Time Granularity in Simulation Models within a Multi-Agent System

Edjard de Souza Mota

Ph.D.
University of Edinburgh
1998

Abstract

The understanding of how processes in natural phenomena interact at different scales of time has been a great challenge for humans. How information is transferred across scales is fundamental if one tries to scale up from finer to coarse levels of granularity. Computer simulation has been a powerful tool to determine the appropriate amount of detail one has to impose when developing simulation models of such phenomena. However, it has proved difficult to represent change at many scales of time and subject to cyclical processes. This issue has received little attention in traditional AI work on temporal reasoning but it becomes important in more complex domains, such as ecological modelling.

Traditionally, models of ecosystems have been developed using imperative languages. Very few of those temporal logic theories have been used for the specification of simulation models in ecology. The aggregation of processes working at different scales of time is difficult (sometimes impossible) to do reliably. The reason is because these processes influence each other, and their functionality does not always scale to other levels. Thus the problems to tackle are representing cyclical and interacting processes at many scales and providing a framework to make the integration of such processes more reliable.

We propose a framework for temporal modelling which allows modellers to represent cyclical and interacting processes at many scales. This theory combines both aspects by means of *modular temporal classes* and an underlying special temporal unification algorithm. To allow integration of different models they are developed as agents with a degree of autonomy in a multi-agent system architecture. This *Ecoagency* framework is evaluated on ecological modelling problems and it is compared to a formal language for describing ecological systems.

Acknowledgements

I would like to express my heartfelt gratitude to my supervisors Dave Robertson and Alan Smaill for their objective advice over the years of my research. Their effort, beyond any duty, helped me to turn this work into a dissertation.

I am most thankful to my examiners, Dr. John Hallan and Prof. Micahel Fisher, for their helpful suggestions and an enjoyable intellectual exercise during the Viva.

I owe a big thank-you to my dear friends Wamberto Vasconcelos for those unforgettable "academic coffees" at "Sorriso" which helped me in the first steps to write down my ideas about time, and Paulo Salles for many patient discussions about ecological models, and Alberto Castro Jr. who revived the "academic coffees" and for sharing stimulating discussions which helped me in the end of this work.

Thanks to Robert Muetzelfeldt who participated in many discussions about the problem of granularity in simulation of models of ecosystems. His help cleared many doubts on this subject.

Thanks are also due to my fellow colleagues of the KBS group, Jessica Chen-Burger, Peter Funk, Nam Seog Park, Jane Hasketh, Steve Polyak and Renaud Lecoeuche. Folks, your patience in listening my presentations and the fruitful discussions helped this work to an extent more than you can realise.

Thanks to Louise Dennis, Paula Lourenço and Steve Polyak who helped with "proof-reading" of my scribbles.

I am most thankful to my family and friends back home who always gave me the support and encouragement necessary to keep my pace. My brothers Edjander and Edjair (in Germany) who always recalled our good moments playing football under the supervision of our best coaches ever - dad and mum. These remembrance lifted my courage to keep me digging in this field with the same enthusiasm as we enjoyed our "peladas".

An enormous Thank-you is due to Cirlene for her presence (and some times absence), support and cooperation without which I would certainly have found the stones of the way a lot heavier.

The friendship of Goretti, Teresa, Susan, Marco Aurélio, Debora, Felipe, Herman and Patricia, Gisela, and Hugo Terashima gave me support in the last year when I most needed. Thanks a lot folks, you made the last year a lot colourful when, in certain moments, I thought it was blue. Recently, new folks brought more colours - thanks José, Daniela, Marcio, Ana and Heloisa.

I would like to thank the financial support of the Brazilian people through the Fundação Coordenação de Aperfeiçoamento de Pessoal de Nível Superior (CAPES) of the Ministry of Education, under grant nº 01723/93-8/CAPES. Thanks are also due to the Universidade do Amazonas for supporting my leave of absence, specially to the staff lectures at the Department of Computer Science for backing me up during this period. I also would like to acknowledge the support provided by the Student Travel Fund of the Department of Artificial Intelligence which enabled me to attend the Workshop on

Executable Temporal Logics along with the IJCAI-95 (Montreal, Canada).

I am extremely grateful to the secretaries Jean Buten, Janet Lee, Karen Konstroffer and Amanda Hearn, not only for their efficient secretarial or administrative support but also for they have been kind when I needed their support. The path of my research was much easier due to the efficient library services of Olga Franks. Computing support is not an easy task, Craig, Mike, Neil and John thanks for doing the business with the most high level of quality.

I would like to express, from the bottom of my heart, my thanks to the people of Scotland represented by my front door neighbour Edith Ferguson. Her kindness, attention, and care made life at St. Leonard's Street full of enjoyable memories. Thanks Edith, you are a truth ambassador of the Scottish warm style of having foreigners living in their lovely country.

I dedicate this work to my parents, Edgard and Conceição for their constant support and love, since my early steps toward education.

Declaration

I hereby declare that I composed this thesis entirely myself and that it describes my own research.

Contents

List of Figures

List of Algorithms

Chapter 1

Introduction

The central issue of this thesis concerns the granularity of action and time. In particular, it is about how to specify and execute interacting processes aligned and/or non-aligned in time, and working at different time scales. Rather than taking a more traditional application domain, I investigate the problem of integrating simulation models of ecosystems where a new approach for tackling the complexity of many processes with such features is particularly necessary.

Usually, such models are specified in programming languages which hide the concepts of actual systems in very complex algorithms. This makes the task of integrating different models a very difficult one. It would be helpful if modellers could use a high level representation framework to be used for building reliable prototypes of considerably complex models of ecosystems. This would help them to test their ideas before they translate them to more efficient architectures. A good side effect of this would be that such high level descriptions could be put on the top of the more efficient ones in order to make integration with others more understandable.

This chapter is devoted to the presentation of the central idea of this thesis starting by a hypothetical example as a motivation for the work. A discussion is followed concerning the possible problems we may come across when trying to tackle the challenge of specifying a computational model to represent and/or simulate the scenario proposed. Then, after suggesting the reasons why traditional approaches do not offer suitable mechanisms to make them more useful, I address the solution this work proposes.

1.1 Motivation

In this section I will present a hypothetical example which deals with the same kind of knowledge as in actual ecosystems and the importance of having ecological models to treat them. It would be extremely difficult to analyse a real example of an actual ecosystem in the space of this work, mainly because it would be necessary a lot of expert knowledge on ecosystems. Also, the complexity of knowledge of actual systems can be approximated by examples such as the one below which raises enough problems to be solved.

> **Example 1** *John* plants a *tree*, and begins observing its growing process every week. After some time an insect pest appears and starts damaging the *tree*'s growing process on a daily scale. *John* decides to use some chemical resource p_j, from *Johnston* chemical producers, against the *bug_demon* which goes away, but some portion of p_j stays on the leaves of the *tree* which absorbs it in a few hours. The *bug_demon* returns after some time and more chemical needs to be sprayed. This occurs during the same period of time every month. After some months *John* meets a friend, *Elza* who works for *Buyern*'s company of chemical resources, and she says p_b from her company is much more efficient against that type of pest and the side effects in the *tree* are less harmful. *Minor*, from a government institution of research says *John* should use both p_j and p_b since it would be better for the environment and for the economy. In the end, *Angelina* appears, from a tribe in the rain forests who suggests that *John* use a natural resource *bug_angel* which attacks the *bug_demon* and its excrement fertilises the soil around the *tree*, but *John* is very skeptical about non scientific knowledge.

In this hypothetical story[1] there are different types of knowledge involved, like the tree's growing process, the influence of: trees upon other trees, bugs on the growth of a tree, chemical substances upon plants, population dynamics of bugs and competition between bug communities. All of these processes are associated with different scales of

[1] Any resemblance with real facts and names will be mere coincidence.

time, repetition, or cyclicity of events. This becomes more complex if we also consider other animals within the environment that could influence these processes. Other very important types of knowledge are the knowledge *John* has about ecology of plants and animals, about products from chemical producers, governmental policies for natural resource management, and how they all should interact to guide his decision on what to do.

Research institutes, universities, companies, industries and governmental institutions have models of the environment in terms of databases, spread sheets, simulation models, etc. The great majority of those outside these communities, can only access these through libraries and the media in general. With the advent of modern computer technologies, such as multimedia and the world wide web (WWW) on the Internet, different forms of media may be used to communicate ecological information directly from governmental and non governmental institutions, universities, etc. Although all of this knowledge is becoming more accessible, most of the computer systems developed to help our understanding of ecological processes and how we interact with them do not combine a wide variety of sources of knowledge. Why does this happen? Possible answers are that:

- they are usually directly specified in terms of complex algorithms, and so it is very difficult to extract the knowledge associated with them;
- they can be modularly developed, but to compose their subcomponents in order to represent a complex ecosystem we are forced into a rigid and invariant structure.
- they are *closed systems*, because they cannot be automatically integrated with other systems, i.e. to adapt them to new components we usually need to reprogram them;
- they do not allow us to freely intervene in the simulation and interact with it as a component of the model. This makes us look at these models as if we were far from the real environment they are supposed to simulate.

The natural consequence is that traditional simulation models of ecosystems are not as helpful as they could be for making predictions. Even worse, they are of limited use in helping us understand how the whole environment works. The problem is that the computational frameworks used do not offer suitable mechanisms for high level descriptions of models working at many scales (of time, space, organisation) which can also be easily integrated with other systems and modellers, or decision makers. Such a high level framework could offer, for example, suitable means for defining human influence (actions) over ecosystems which allow us to actually intervene in parts of/and during the simulation in an analogy to our action over actual ecosystems. Note that this does not depend only on friendly graphical interfaces, but also in considering human action as a more active part of the simulation. We now consider the problems with such systems in more detail.

1.2 Why Traditional Approaches Fail

Traditionally, most of the scientific effort for modelling physical systems in computer programs has been to subdivide all of the knowledge involved, and then to investigate each part of it separately by means of particular frameworks for each domain. Because of this, different assumptions are made and conclusions on what to do are usually guided by urgent decisions and interests of those who have more information (not necessarily knowledge, which explains their decision). Unsurprisingly, such information does not always reflect the common sense of all those people involved. Yet, it is impractical to model the whole environment.

Although the development of computer systems has provided many tools for automating much of the inference involved in situations like that of **Example 1**, the problem is that each computer system, based on complex mathematical models, assumes specific scales of time, space, and organisation. However, for each level, many assumptions are made that usually do not take into account the influence, through the flow of time, of other levels. As a result, the *scaling up* of individuals and localised processes to the whole environment seems to be nearly impossible. The same "human phenomenon" seems to occur when a decision about how to act upon the environment is made based only on local knowledge about the target of the action.

A simple way of integrating ecological models is to prescribe how different models should pass information to one another. This practice does not impose any discipline taking properties of the whole environment into account when defining sub-models of it. Furthermore, there is no guarantee that for any new piece of information the overall model would not have to be updated. A modular way of describing the overall ecosystem in terms of its parts, and how they relate to each other, is necessary from this perspective rather than an efficient programming language. However it is necessary that the language should be efficient enough to meet the demands placed upon it. In order to offer these desirable features, such systems

- should provide more general ways of interconnecting the knowledge embodied within them,
- should be *open systems*, i.e. be able to react either when new components are introduced in the computer environment in which they are running or are moved from it, with minimal effort of computer programming,
- should be able to interact with others specified at different scales of time, space, structure, etc.
- should represent the concurrent and parallel aspects of interactions between entities in ecosystems.

The paradigm which is most likely to supply these features is the *multi-agent system* (MAS). What makes MAS a good paradigm for representing models of ecosystems rather than traditional simulation models? Representing processes of an ecosystem in terms of interacting agents offers a better computational building block to compose complex ecosystems from its subparts. The reason is because each agent can embody its own picture about the environment and so the effort of the modeller is left to investigate properties of agents, interactions between agents. This will help modellers test their theories about how things, which happen at lower levels of granularity, will affect the others at higher levels.

1.3 Why OOP is not Enough

In general, an agent can be seen as a specialised object, and so one can use the object oriented paradigm (OOP) to build an agent architecture. I chose not to use the OOP as a specification language because I did not want go into a deep chaining of class specifications that may not be relevant to the problem itself. For instance, in the *Eco-Talk* system [Baveco & Smeulders 94], the modeller has to worry about many details of class specifications that are out of the scope of the problem, *e.g.* this system provides classes like *Experiment* and *Simulator*. Some other very interesting *ad hoc* models based on OOP are proposed in [Makela *et al.* 93], [Baveco & Lingeman 92], [Folse *et al.* 89], but these involve similar problems. At best, they provide some libraries of classes suitable for their domain of application.

A good attempt to provide a more general framework for ecosystems based on OOP is the ECOWIN approach [Ferreira 95] which provides classes to represent important elements for the specific domain it is applied to, such as *Man* and *catastrophes*. However, its definition for the former is not based on any special human feature and can be really seen as just a specialisation of the latter, mainly because both have methods which cause damage to the environment being modelled. The author addresses the limitations of the approach supporting my argument: "extensions and modifications of models usually require a profound understanding of the code. [This means that] user-based models are frequently very restricted, which does little to diversify the knowledge sources of different processes involved". Thus more complex models cannot be easily handled.

It is worth mentioning the Swarm system, [Hiebeler 94, Burkhart 94], which is also based on an ALife approach. This is a general-purpose package for simulation of concurrent distributed artificial worlds. A swarm is any structured collection of interacting agents, where an agent is defined by a data structure containing its state variables, a step function called at each time step, and action functions to handle messages to the agent by other objects. Its model of time is similar to that used in discrete event simulation. This means that no explicit representation of time granularity or cyclicity is provided.

A more fundamental methodological difference between Swarm and the architecture given in this thesis is that the latter is heavily influenced by declarative description, in a logic programming style. Thus, it employs a different paradigm for modelling, although the ultimate aims of both architectures are similar.

The strong notion of MAS approach, [Wooldridge & Jennings 95], contains a limited set of concepts, which are powerful enough for the needs of simulation models of ecosystems, whereas OOP is a generic programming style and it would be necessary to add such concepts. Shoham, [Shoham 93], pointed out some differences between the elements of both approaches. In Table 1.1, I reproduce this and add some new features which have been proposed in recent years, for example [Cohen & Levesque 90, Singh 94, Haddadi 95]. Note that inheritance is left out because, in the general sense, it is not an essential concept of OOP. The essential thing to compare here is encapsulation which in MAS has a richer (though constrained) vocabulary.

Approach	OOP	MAS (Strong view)
Building Block (BB)	Object	Agent
BB Parameter for state definition	unconstrained	knowledge, capabilities, goals, beliefs, desires, intentions, know-how.
Computation	Message passing and response methods	message passing and response methods
Types of message	unconstrained	inform, request, offer, query, report, deny, accept, reject, command.
Constraints on methods	none	honesty, solidarity, benevolence

Table 1.1: Relation between OOP and Strong view of Agency.

A more ambitious attempt to achieve more general classes of ecosystems is LAGER[2] [Olson & Sequeira 95]. This work uses OOP to provide a platform for building ecosystems using the Artificial Life (ALife) approach. An ALife programme is a set of self-contained abstract "machines" with the full specification for the behaviour of an agent, and there is no representation of the central symbolic world model. Actually, this idea is not very different from MAS approach if we consider ALife as a *reactive architecture* as defined in [Wooldridge & Jennings 95]. The authors claim that OOP

[2] LAGER stands for Large-scale Automated Generic Ecosystem Replicator.

has evolved enough to make the development of such systems an easy task. The problem is that we may fall again into "a road of unconstrained tools" with little hope for integrating different models. The reason is because there is no uniform framework to define models of ecosystems using OOP which offer easy integration of new models.

Along with these problems discussed above, simulation modelling activity could be more effective if modellers could abstract from the elementary instructions of programming language and use special languages with the appropriate constructions to define their models in a declarative way. This raises three issues related to the use of a logic-based framework, and we shall discuss it now.

The first issue is *i*)Why use logic anyway? As it is desirable to write computer programs whose behaviour is a function of the meaning of the structures they manipulate and this can be influenced only by the form of such structures, then we need a systematic relationship between form and meaning [Moore 95]. If we want to unpack some useful knowledge from code of simulation models we need a framework with a clear relation between notation and semantics. Then, this knowledge can be expressed in a declarative way so that it can be easily accessible to modellers. Logic has all this features as was shown in the Eco-Logic project [Robertson *et al.* 91].

This project was a first attempt to explore extensively the use of logic as a high level description tool for improving users' comprehension of simulation models. In such a high level notation, users' perspectives of the problem can be better represented and linked to the solution of the problem by means of a simulation model. However, Eco-Logic was concerned less with the diagnosis of simulation and more with declarative specification of the model. We show a meta-interpreter in a similar logic-programming style as Eco-Logic but which uses special terms to represent temporal entities. We shall see in this thesis that this is an advance on the Eco-logic approach.

From this comes the second issue: *ii*) if it is possible that Eco-Logic approach could be extended to that, then why introduce levels of granularity and cyclicity? The reason is because it is necessary to provide inference mechanisms for the problem of integrating different interacting models and working at different time granularities. For these reasons we decided to develop a logic based framework endowed with the features

lacking in traditional approaches. Why invest in another logic approach instead of extending the existing ones?

First, the known works (see related work on Chapter 2 and Chapter 5) do not seem to be easily extendible to offer such mechanisms. The use of OOP or even of some MAS framework (for example *agent-oriented programming* [Shoham 93] or Concurrent METATEM [Fisher 94a]) does not offer a more direct way of combining models at different levels of granularity. Second, in the case of the existing languages for programming agents they are too general to be of direct use for prototyping complex models of ecosystems which can be executed, though we are not concerned with the efficiency of such models.

Finally, the third issue concerning our research is *iii*) why use agents? Models of ecosystems need to be endowed with their own notion of time, local environment and potential influences so that they may be able to integrate with others. Along with these new features mentioned above (I introduce into the application of logic to ecological modelling), I propose a Distributed Artificial Intelligence (DAI) approach to representation and mechanisms of execution. Logic-based DAI approaches can be seen as better alternatives to the application of standard logic because of the well known controversial applications of the latter (discussed by Moore *op. cit.*). These are, a method of reasoning and as a programming language.

This alternative was not detected by the Eco-Logic project as a candidate for helping in the process of integrating and building complex prototypes of high level simulation models. The good side effect of it, as we shall see, is a considerable improvement in the performance of execution a medium-to-large scale simulation. The research presented in this thesis rescue the hopes that the good features of a logic approach can still be used for building reliable prototypes of models of complex ecosystems in a declarative way.

1.4 Research Questions

Considering that a high level specification framework is demanded for prototyping highly integrated models of complex ecosystems working at many scales of time, we

then consider what sorts of things should be investigated. Assuming that the logic based framework we start with is logic programming for its well known clear relation between specification and semantics, the research questions this thesis intends to answer are:

- Is there any standard pattern of reasoning for combining models? If so, can such patterns be represented formally within a framework for some type of system development?
- Do these patterns scale up from simple to complex examples by making use of the modularity of specifications in logical languages? For instance, specifying clauses for representing coarse levels using clauses which represent finer levels.
- To what extent would it help to separate temporal parts of definitions from other parts of clauses when representing the time dependent and non dependent properties of ecosystems?
- Is there a pattern for time scale definitions which more "naturally" represents the temporal phenomena in nature?
- How does temporal scale interact with structural scale when developing (maybe) complex models at higher granularities to interact with (possibly) simple models at finer granularities?

1.5 Aims of the Thesis

The main goals of this research are:

- to present a time granularity theory which can be easily integrated within a multi-agent architecture, so that models can be specified in their own time scale but easy to integrate with others;
- to provide a standard way to specify processes at different scales associated to their organisation scale, so that agents and groups of agents may present the same pattern of specification.

- to develop a framework for *agent-based systems* (ABS) able to represent an agent with temporal reasoning capabilities, with a notion of local environment;
- to investigate the suitability of using an agent based approach for fast prototyping of simulation models in terms of expressive power and scalability of the system for large scale simulation.

1.6 Overview of the Thesis

Chapter 2 - I shall address the challenges one has to face when trying to represent simulation models of ecological systems (also named as agents throughout this work) by using a temporal representation and reasoning framework. The discussion shows that a certain class of problems, in this domain, can be represented in a particular form of clause for representing deterministic simulations and take advantage of it for representing interacting agents. The limitations of this clause scheme are presented. The chapter will be concluded with a review of the work done so far on temporal (logic) representation and reasoning for dealing with the matters this thesis addresses.

Chapter 3 - We shall see a brief philosophical discussion which suggests that cyclicity and linearity of time can be modelled in the same framework by means of a hierarchy of modular sets. Such a hierarchy offers a natural way to represent different scales (or grains) of time. In this way, models of observed phenomena in nature can approximate the continuous feature of the latter by means of a limited hierarchy of modular sets. **NatureTime** is the logic I propose to represent such concepts. This main idea is to separate the task of defining relations within the model from the task of computing when such relations hold. In this sense, the language can be understood as a quantified temporal logic because it suits simulation models of ecosystems better than temporal modal logic. Finally, we discuss the expressive power and limitations of **NatureTime**.

Chapter 4 - In this work, reasoning about time is considered as a matter of temporal unification. We shall see that temporal unification handles labels and terms related to the temporal universe in a special way. However, instead of having to

test for equality between such expressions a test for unifiability between them shall be used. I will describe a special *temporal substitutional framework* to representing such special terms and a meta-interpreter for **NatureTime** logic. Then, I shall extend it with mechanisms for dealing with interacting agents at different scales of time. Finally, we shall discuss the suitability of this approach for large scale simulation and also point out the reasons why a multi-agent system framework is demanded.

Chapter 5 - I shall propose a conceptualisation of simulation models of ecosystems in terms of agents in a distributed architecture. Such agency is based on the *ecoagent* computational entity. Such an entity is endowed with a degree of autonomy and ability to react according to the influence other agents may exert upon their behaviour. The potential sources of influences over an agent is part of its knowledge and it is a scheme of rules to derive relations only when certain conditions hold. From this an *ecoagent* keeps its local environment updated whenever the environment changes. I will then evaluate *Ecoagency* framework on ecological modelling problems and compare to a formal language for describing ecological systems. We do not consider our approach as deliberative in the usual meaning of this term. Neither it is reactive, and we explain this by placing *ecoagency* within the recent researches on distributed artificial intelligence.

Chapter 6 - In this chapter, we shall see one way of implementing the concepts related to *Ecoagency* framework. The multi-agent architecture specified allows one to represent and execute models of ecosystems with a degree of complexity which would not be feasible to be executed in standard computational logic, and even in **NatureTime**. In the end some important properties of the system will be demonstrated.

Chapter 7 - I will show some experiments with the architecture described in Chapter 6 applied to the modelling of ecosystems. The examples we shall see have the purpose of investigating the suitability of using our approach for building medium to large prototypes of simulation models with the same level of complexity of typical ecological models. We also use them to evaluate the expressive power and limitations of the architecture, and in the end we shall see what could have

been done to overcome the limitations of the present architecture.

Chapter 8 - I shall address the conclusions of this work and the contributions I claim this research has achieved. The chapter will finish with a brief discussion about the goal to make simulation models accessible to people who must use the models to make decisions (or to teach, etc.). In this discussion, we shall see the contributions of this thesis in relation to such a long term enterprise and I propose some further work to make it a little closer than it has reached so far.

Chapter 2

A Challenge for Temporal Logic

... the forces governing life history evolution, shaped by competitive pressures and co-evolutionary interactions, are such that each species observes the environment on its own unique suite of scales of space and time.

Simon A. Levin (The problem of Pattern and scale in ecology. Ecology(73), 1993).

2.1 Introduction

In the last two decades theories for temporal (logic) representation and reasoning have been developed dealing with a wide class of problems from different areas of application. These areas, such as natural language, temporal databases, planning, reactive systems have not been much concerned with the specification of interacting processes at different time scales. Such processes, as we pointed out earlier, may also be cyclic and be aligned or non-aligned through the flow of time. Although some attention has been paid to parts of this problem, there has been little attempt to address the wider issue of interaction between different temporal properties.

This chapter explains how modelling ecological systems challenges temporal representation and reasoning according to the features given above. I will show why a particular form of clause for representing deterministic simulations, which yields courses of values, may be interesting for representing interacting agents. I will also indicate the reasons why the simple use of such a clause schema is not enough to provide the desirable features that could make systems of this type more useful, as I proposed in Section 1.2. The chapter concludes with a review of the work so far on temporal (logic)

representation and reasoning for dealing with these matters.

2.2 Simulation Clauses

A large class of ecological simulation models can be represented using a standard clause schema which we call a *simulation clause*. Usually the specification of a simulation model to compute the value, V, of an attribute, Att, of a given agent, Ag, at a time, P after T is in the form

$value(Att, Ag, V_i)$ @ T_i.

$(value(Att, Ag, V)$ @ $(P\ after\ T_p)) \Longleftarrow (value(Att, Ag, V_p)$ @ T_p & $\mathcal{R}(V_p, V))$.

where the first clause states that the value of the attribute Att of Ag is V_i at the initial time T_i. In the second clause, P is a length of time of size 1 at some scale; $after$ is a displacement function from T_p to the time P units later; and $\mathcal{R}(V_p, V)$ represents a sequence of formulae which imposes constraints on the clause (including recursive subgoals which may be for values of the attributes of other agents, but not for Att of Ag) and calculate a value for V from that of V_p. So, the first expression says the value of the attribute is V at time P after T_p if (represented by $\Longleftarrow$) the conditions in the body hold.

Note that the constraint on the existence of only one recursive subgoal relating Att of Ag establishes the threshold of the class of models of ecosystems we are not concerned with. The main reason is because a large class of physical phenomena such as those related to ecosystems usually does not need to represent state transition taking two past states into account. If one wants to extend the expressive power of the scheme it has to be concerned with possible problems which may appear such as incompleteness.

One advantage of this representation is that temporal knowledge is separated from knowledge about the state of the world. Standard computational logic does not allow this, *e.g.* [Robertson *et al.* 91], and so we cannot make general claims about temporal assertions. Also, a simulation model can be expressed as a relation between the state of the agent in the past (or present) and its state in the future, given certain conditions of state transition. This means that specifications of this kind can be executed to

construct a model to satisfy a future state of the world. This is similar to the idea of *declarative past and imperative future* [Gabbay 89].

Another advantage is that to reason with simulation clauses we simply follow a single chain of conclusions until we have reached the desired time. Basically, we start from a known value for *Att* of *Ag* at some initial time. Then, given a known final time at which we want to know the value of *Att*, we match the sub-term $value(Att, Ag, V_p)$ @ T_p on the initial value and recurse on the head – provided that $\mathcal{R}(V_p, V)$ holds. This work assumes a forward chaining strategy for simulation clauses.

The simulation of the agent's behaviour through the flow of time corresponds to computing or running the program for a temporal sentence, at each time step, until the solution is found. A temporal query is a goal in which we want to know if some relation holds at a given time, or throughout an interval of time. The term "computation of a solution" has the same meaning as in [Hogger 84] with the difference that we extend it for temporal formula.

In what follows I show the problems we have by using such a clause scheme in the representation of interacting agents.

2.3 Agents Interacting

Consider the example of Section 1.1. Let's explore three possible ways to propose models about some of its components.

> **Example 2** "There are models of tree growth, t_1 and t_2, expressed on a weekly time scale and t_1 influences the growth rate of t_2 and vice versa. Both models are aligned in time, i.e. the update of the height of each tree they represent happens at the same time".

In this example only one scale of time is necessary, and so the elements of the language can be just basic time intervals along with temporal operators, *e.g.* $after$. Using the simulation clause schemata their specification could look as follows, where the initial tree heights are assumed to be 5 and 7 meters, respectively; C_1 and C_2 represent

constraints or properties which must hold according to the values of H_1 and H_2; they are used to compute H_f.

$value(height, t_1, 5)$ @ 0.

$value(height, t_1, H_f)$ @ 1 $after$ $T_p \Longleftarrow$

$value(height, t_1, H_1)$ @ T_p &

$value(height, t_2, H_2)$ @ T_p &

$C_1(H_1, H_2, H_f)$.

$value(height, t_2, 7)$ @ 0.

$value(height, t_2, H_f)$ @ 1 $after$ $T_p \Longleftarrow$

$value(height, t_2, H_2)$ @ T_p &

$value(height, t_1, H_1)$ @ T_p &

$C_2(H_1, H_2, H_f)$.

The inference mechanism can be as simple as the Prolog meta-interpreter presented in [Sterling & Shapiro 86], but with a different search strategy, i.e. forward chaining. However, this solution cannot be used if another agent working at a different scale of time is introduced, unless we flatten all the temporal scales to a single one.

> **Example 3** "There is a model of tree growth t_1, but now it interacts with a model of an insect pest, say b_1 which moves up and down on a daily time scale, reversing direction when it reaches the top. t_1 has its growth rate reduced by some value every time the pest moves up to the level of t_1's canopy of leaves."

Now, temporal language needs to represent scales. We might then assume a hierarchy of scales, and the inference mechanism should be able to reason about aligned processes working at different time scales. The models could look like as follows, where $t(1,1,1)$ stands for *first day of the first month of the first year*; $p(1, C)$ stands for *period of 1 unit of scale C*; *altitude* represents the vertical position of the bug, and $S...\lfloor T \rfloor$ represents a linear interval open on the right, i.e. T is not included; C_1 represent the constraints on t_1's reaction to the attack of b_1; C_b represent the behaviour of b_1 according to the height of t_1; $included(T, I)$ is true if T is a time included in interval I.

$value(height, t_1, 5)$ @ $t(1,1,1)$.

$value(height, t_1, H_f)$ @ $p(1, week)$ $after$ $T_p \Longleftarrow$

$value(height, t_1, H_1)$ @ T_p &

$P = [V \mid value(altitude, b_1, V)$ @ T and $included(T, T_p ... \lfloor p(1, week)$ $after$ $T_p \rfloor)]$ &

$C_1(H_1, P, H_f)$.

$value(altitude, b_1, 4)$ @ $t(1,1,1)$.

$value(altitude, b_1, A_f)$ @ $p(1, day)$ $after$ $T_p \Longleftarrow$

$value(altitude, b_1, A_p)$ @ T_p &

$value(height, t_1, H_1)$ @ T_p &

$C_b(A_p, H_1, A_f)$.

In this example a representation for non-aligned agents is not necessary because both models will always update their state at the same time, in spite of one being faster than the other. Note that in the specification of t_1 a list notation is used to represent the values of b_1's altitude during one week, and the constraints to form this set are simple because both agents are aligned in time. Complications appear if we need to represent non-aligned processes.

> **Example 4** "There are two models of trees, t_1 and t_2, as before, but now t_1 and t_2 are non-aligned, i.e. they do not have their attributes updated at the same time. Both models also interact with a model of an insect pest which moves up and down on a daily time scale, and horizontally every week. While t_1 has its growth rate reduced by 0.02 every day the pest moves above 8 meters, t_2 growth rate is reduced by 0.01 when the pest is below 9 meters. The pest moves continuously up and down, at a rate of 2 meters per day, reversing direction when it reaches the top or bottom. It moves horizontally either when it has enough of one tree and chooses another one, or in the case a malentity attacks it."

In this last situation, the elements of the language are the same but the inference mechanisms must deal with non-aligned processes. Because of this, the interaction between t_1 and t_2 will result in a list of values for both. The model might be as follows.

$value(height, t_1, 5)$ @ $t(1,1,1)$.

$value(height, t_1, H_f)$ @ $p(1, week)$ $after$ $T_p \Longleftarrow$

$value(height, t_1, H_1)$ @ T_p &

$P_1 = [V \mid value(altitude, b_1, V_1)$ @ T and $included(T, T_p ... \lfloor p(1, week)$ $after$ $T_p \rfloor)]$ &

$P_2 = [V \mid value(altitude, t_2, V_2)$ @ T and $included(T, T_p ... \lfloor p(1, week)$ $after$ $T_p \rfloor)]$ &

$C_1(H_1, P_1, P_2, H_f)$.

$value(height, t_2, 7)$ @ $t(10,1,1)$.

$value(height, t_2, H_f)$ @ $p(1, week)$ $after$ $T_p \Longleftarrow$

$value(height, t_1, H_1)$ @ T_p &

$P_1 = [V \mid value(altitude, t_1, V_1)$ @ T and $included(T, T_p ... \lfloor p(1, week)$ $after$ $T_p \rfloor)]$ &

$P_2 = \{V \mid value(altitude, b_1, V_2)$ @ T and $included(T, T_p ... \lfloor p(1, week)$ $after$ $T_p \rfloor)]$ &

$C_2(H_1, P_1, P_2, H_f)$.

$value(altitude, b_1, 4)$ @ $t(1,1,1)$.

$value(altitude, b_1, A_f)$ @ $p(1, day)$ $after$ $T_p \Longleftarrow$

$value(altitude, b_1, A_p)$ @ T_p &

$value(height, t_1, H_1)$ @ T_p &

$value(height, t_2, H_2)$ @ T_p &

$C_b(A_p, H_1, H_2, A_f)$.

Note that, syntactically, there is little difference between the specification above and one made by using standard computational logic. The actual difference lies in the mechanism of inference for dealing with non-aligned processes (and possibly cyclical processes). The Table 2.1 presents the relations between the time scale of the interactions between agents, the temporal elements necessary to represent such interactions and the inference methods we need to treat them.

Interacting Agents	Temporal Elements	Inference
Single time scale	Basic time intervals + $after$ ($before$) function	Basic meta-interpreter
Hierarchical time, aligned	Time hierarchy + Pure Temporal Expressions	Aligned, hierarchical meta-interpreter
Hierarchical time, non-aligned	As above	Non-aligned, hierarchical meta-interpreter

Table 2.1: Interacting agents, representation and inferences needed.

2.4 The Problems this Work Address

There are two main issues to be considered in these three examples. The first concerns the specification of interacting aligned and non-aligned processes at different scales of time. In **Example 4** there are three interacting entities working at different time scales. The mechanisms of inference should cope with the following questions:

i - "what is the value of the attribute of each agent at a specific time in between two consecutive time steps at the agent's scale of time?"

ii - "what are the values of an agent's attribute during a certain period of time? (for aligned and non-aligned periods in relation to changes on the agent's attributes)".

I assume that the state of such agents changes at the time step of their corresponding granularity. One might argue that a more "realistic" representation would be the use of differential equations, which is an expressive way of representing the *continuity* of their behaviour. However, a continuous representation is very hard to solve automatically. Although continuous and discrete approaches may produce results which are numerically different, they are qualitatively similar to one another [Gotelli 95]. Moreover, many continuous models, in practice, use a discrete approach to approximate results. I chose the same line and only discrete models will be treated in this work.

The second issue is how the first one is affected by the composition of different models to form complex models, so that we may obtain consistent integration of them. Suppose we start with a model for **Example 2**. Then, another modeller wants to introduce a model at a different time scale, we then progress to **Example 3**. Finally, a third person may wish to introduce other models with processes non-aligned to the existing ones, and we reach **Example 4**. We could make things more complicated if we decided to introduce a model to represent the action of a pesticide, water-soil, or any other non-expected events that may suddenly change the state of an agent.

Of course the models of the examples above do not represent, in general terms, the influences on the attribute of an agent. For this it is necessary to find a way to represent all possible influences over an agent, and then obtain the progress of such influences during the period of time the agent changes its state. To do this I propose

the following concept, where $influence(Att_i, B, Att_j, A)$ represents that *attribute* Att_i *of agent* B *influences attribute* Att_j *of agent* A, and $\mathcal{R}(V_p, V_L, V_f)$ represents a sequence of sub-goals which relates V_p, V_L and V_f.

Definition 2.1 (General Simulation Clause Scheme) *Let Att be an attribute of an agent A which changes every P units of time, and* Att_i *be an attribute of another agent B which affects A. The* general simulation clause scheme *(GSCS) of A's behaviour which changes Att is*

$value(Att, A, V_i)$ @ T_i.

$value(Att, A, V_f)$ @ P *after* $T_p \Longleftarrow$

$value(Att, A, V_p)$ @ T_p &

$I = \{(Att_i, B) \mid B$ *is an agent and* $influence(Att_i, B, Att, A)$ *holds* $\}$ &

$V_L = \{(Att_i, B, [V \mid value(Att_i, B, V)$ @ T *and* $included(T, T_p ... \lfloor P\ after\ T_p \rfloor)])$

$\mid (Att_i, B) \in I\}$ &

$\mathcal{R}(V_p, V_L, V_f)$.

where $value(Att, At, V_i)$ @ T_i *is the initial state of Att.*

However, when representing agents by means of GSCS it has to be assumed, by whoever is going to use it, that such agents have other inference capabilities along with the ones necessary for dealing with granularity of time. Depending on which kind of agent and attribute are being represented, such capabilities may be constrained by human-like notions or not. These can be computational notions of beliefs, desires and intentions or whatever better suits the application domain.

In the rest of this chapter I will review the current theories for temporal (logic) representation and reasoning which could be thought as suitable frameworks for tackling these problems.

2.5 A Survey on the Granularity and Cyclicity of Time

2.5.1 Articulating Time Theories

In our daily life we face phenomena happening at different scales of space and time. We have developed the ability to perceive how things which occur at finer-grain levels relate to those which are more coarse-grained. What is most interesting is that we seem to automatically abstract detailed information, when the effects of it at higher levels is more relevant. Of course we start losing this understanding when the scale and the number of agents and interactions among agents involved in the phenomenon increases. To our knowledge, the first work to suggest that a machine endowed with moderate intelligence should have a theory of granularity was [Hobbs 85]. This work proposes that granularity can be obtained by mapping a global theory of time, called $\mathbf{T}_0$, into a more "coarse-grained" theory $\mathbf{T}_1$.

Each coarse-grained theory, called local theory, collapses the complex one from which it was built up. A theory of granularity would be formed by a set of local theories and how they *articulate* (or relate) to each other. No formal representation was proposed for connecting local theories and how they relate to the flow of time, and so no inference rule could be devised about relations between local theories. Despite this, some very important notions were introduced which would influence later work.

One important concept proposed is the notion of an *indistinguishability* relation to delineate the scope of validity of local theories. For instance, the principles and processes associated with a forest can be applied to any piece of the forest and it will still be a forest. But when we move down to a finer level, *e.g.* trees, the internal processes of a single tree are not identical to those of the forest it belongs to. In this sense, local theories at the forest level do not hold at tree level. However, from the organisational point of view if we compare the behaviour of an individual tree (or animal) with the behaviour of a forest (or a population of animals), we may find similarities in the way they behave. The computational model I propose in this work allows us to model both levels in a similar way.

2.5.2 Time Units of Real Clocks

Looking for a way to represent real clock time at many scales, [Ladkin 86b] extended the interval calculus [Allen 83]. Allen's calculus has as the primary temporal object a *convex* interval, an interval with no gaps. Because of this feature it is not possible to represent intermittent processes. Ladkin's approach introduced the notion of a *union-of-convex* intervals (UoCIs). By using such intervals it is possible to define different time units, which are sequences of integers associated to sorts [Ladkin 86a]. This system was called the Time Unit System (TUS).

The completeness of TUS was demonstrated in [Ladkin 87], where Ladkin showed that it contains a canonical model for the interval calculus. More recently, [Bouzid & Ladkin 95] proposed a temporal ontology of UoCIs represented in the TUS, and they addressed the advantages of a reified logic over a non-reified one.

Along with the problem of combinatorial explosion of non-convex intervals (as Ladkin showed), the claimed "natural representation" of time clocks does not seem to match the nature of "natural clocks". Natural clocks are inherently cyclical. Thus time should not be represented as a pure linear structure but combine linearity with cyclicity. Besides the lack of a proper approach to this philosophical issue the pragmatics of using TUS for computation are, at least so far, obscure.

2.5.3 Granularity of Calendars

In [Leban *et al.* 86], Leban *et. al.* proposed a recursive way for building up a collection of intervals from an infinite contiguous sequence of intervals called *calendars*. The basic idea is to use a set of primitive collections to specify other collections by using *slicing* and *dicing* operators. Such operators allow the selection of intervals from collections of intervals. Each primitive is defined by specifying the intervals of which it is composed.

In this approach, circular aspects of time can be obtained from the *δ-values* which are treated as if they were a circular list. Although this framework was shown to be useful for reasoning about scheduling, it does not really deal with different granularities of time, and it does not allow the specification of relationships between propositions defined over different time scales.

2.5.4 Granularity of Time in Temporal Databases

A pragmatic approach for dealing with time granularity was proposed in [Dean 89], in order to speed up the information retrieval on a large temporal data base. The hierarchical framework of time scales, called a *time map system*, uses a structure similar to the usual calendars. In this way, information at temporal levels of abstraction can be represented and retrieved.

The hierarchy over a linear structure of time is obtained by the concept of a *partitioning scheme*. This is a sequence of partitions $P_1, P_2, \ldots, P_n$ of the set of reals, in such a way that for each $i < n$ if an interval I belongs to a partition P_i, then there is a set of time intervals in P_{i+1} such that I is partitioned by it. Although this approach appears to be useful in the context of data base maintenance, representation and reasoning about cyclical events does not seem to be possible. The main reason is that variables in temporal queries are interpreted as existentially quantified rather than universally. Universal quantification is the mechanism used by systems based on a linear view of time to represent intermittent processes.

Furthermore, it does not offer a mechanism for prediction through time, although it might be possible to build one. Since we often want to obtain some prediction via inference rules, and not only retrieve assertions about temporal knowledge, then this approach is not suitable for representing our simulation models.

2.5.5 Temporal Granularity via Topological Logic

Topological logic, [Rescher & Urquhart 71], extends standard propositional logic with a parameter operator P_α. The meaning of $P_\alpha(p)$ is that "proposition p is realised at the position α". We can use this framework to reason about knowledge which is temporally and/or spatially dependent.

[Ciapessoni *et al.* 93, Montanari 94] proposed a many-sorted first order predicate calculus within a topological logic augmented with *contextual* and *projection* operations. The first identifies the domain or level of time granularity at which a given formula has to be considered. The second is used to constrain formulae to different domains. The gains from this "wedding" are the use of temporal operators and a metric on time

to deal with time granularity. The hierarchy of time is a linear structure called the universe of domains, where a *granularity ordering* relationship is imposed over this universe. A a partial ordering of *disjointedness* is also provided to relate domains at different levels of granularity.

This is similar to the *partitioning scheme* of Dean's approach and to the local theories, suggested by Hobbs (*op. cit.*). Temporal domains are related to our concept of *modular temporal class*. Their concept of locally temporally valid is related to the meaning of the throughout temporal connective. Because we define grains of time based on chain of modular sets, the cyclicity of events happening at different granularities is more easily obtained. Moreover, there is no mechanism for dealing with non-aligned interacting agents at different time scales.

2.5.6 Granularity in the Decomposition of Abstract Actions

For specification and reasoning about reactive systems, [Fiadeiro & Maibaum 94] proposed a hierarchical (vertical) decomposition (or abstract implementation), of object specifications in temporal logic. Such objects are seen as building blocks of a hierarchical reactive system design. At each layer of the hierarchy there is a logic dealing with a single time scale, isomorphic to the set of natural numbers, and there is a collection of objects that may be used for composing complex objects (systems) at higher levels of abstraction.

In this way, temporal execution of an abstract action is done by the temporal execution of concrete actions of the level below. The interface between both levels is given by axioms which say when the concrete actions start, are being executed or have finished. This work does not intend explicitly to represent or reason about time. However, closer observation shows us that the granularity of time is embodied within the specification of actions at each level. It is not clear how cyclical processes might be represented in such a framework. Although we are allowed to represent interaction between abstract and concrete actions, the opposite direction of the relation does seem to be straightforward. The approach I chose, though based on explicit reference of time and different mechanisms, is more general because interaction is allowed in both directions.

2.5.7 Time Granularity via Local Clocks

In [Fisher 96] a mechanism for a specification with many granularities within the same logic is provided. In this work, granularity (though not mentioned) is achieved by providing each agent with its own local clock represented by the predicate $tick(O)$. To avoid the problems of committing the logic to have a "next" operator fixed to a single (multi purpose) granularity, Fisher used auxiliary predicates $next\text{-}tick(O, X)$ - which is true if X is satisfied within the next tick O, and analogously $last\text{-}tick(O, X)$.

In this work, agents execute temporal specifications in a discrete model of time. As a consequence each of this agents construct its own temporal model. To coordinate asynchronous executions, however, a dense temporal logic is used to describe the semantics of groups of agents. This means that local ticks of the local clocks are flatten to a (dense) linear sequence of global ticks (or global clock).

Granularity of action is related to agent's state which corresponds to tick intervals. Because there can be different tick intervals, or state transition at the object level, there may exist overlap of execution at the overall system level. This overlap of execution corresponds to non-aligned periods of change of the agent's attributes (see Section 2.4). Despite this solution, no mechanism is proposed for dealing with interacting agents working at different ticks of the global clock. The mechanisms we could expect are related to the kinds of inference one agent at one level of abstraction is able to perform about its own properties which others, at other levels, may access or even influence or be influenced by.

2.5.8 Granularity in Temporal Logic Programming

Recently, [Liu & Orgun 96] proposed an extension of Chronolog [Orgun & Wadge 92] to deal with multiple granularity of time, called Chronolog(MC). While in Chronolog all predicates are defined on the global clock, Chronolog(MC) is based on a temporal logic with clocks in which we can associate a local clock with each predicate symbol. Every program in this language is composed of a) a clock definition, b) a clock assignment and c) a program body. In this way every predicate symbol which appears in (c) is associated with a local clock through (a) and (b).

A clock ranges over natural numbers $\mathbb{N}$ and it is defined by a special predicate $cki/1$, where i is the identification a particular local clock. A local clock is specified as follows, where $first$ and $next$ are prefix temporal operators, E is a single-valued function from $\mathbb{N}$ to $\mathbb{N}$ which also defines the grain or scale of the clock (a pure number).

$first\ cki(n)$.
$next\ cki(N) \leftarrow cki(M),\ N\ is\ E(M),\ N > M$.

A clock assignment is simply done by a sequence of facts of the form $is\text{–}ck(p, cki)$, where p is a predicate symbol and cki a local clock. After this, a program is written using Chronolog syntax which uses the same temporal operators $first$ and $next$. However, because of the clock assignment the interpretation of each predicate is according to the local clock associated with it.

The execution of Chronolog programs is based on a clocked TiSLD-resolution proof procedure. Basically, whenever a clocked temporal atom is selected from a goal, it is matched against program clauses and a special unification algorithm performs the matching of clocks. The successive application of such a rule guarantees to finds a solution which is part of the minimum clocked temporal Herbrand model, if there is one.

Chronolog(MC) has been applied in the specification of simulation systems, where (c) is used for describing the functionality of each process within the system; (a) and (b) are used to describe temporal properties related to the behaviour of the processes. The authors claim that the granularity of time is more flexibly represented because there is no commitment that a grain of time is a refinement of another one. Depending on the application domain it might be true that such a flexibility can be an advantage. However, for problems where there exists interacting processes working at different clocks this can be a problem.

Another limitation of Chronolog(MC) is that it cannot be used to specify processes in the style of simulation clauses as defined in Section 2.2. The reason is that it is not possible to define "clocks in terms of the same or other clocks", as Liu and Orgun pointed out late.

2.5.9 Granularity and Cyclicity Combined

In [Cukierman & Delgrand 95] a framework for representing calendars is proposed. The basic idea is to consider *calendars* as cyclic temporal objects. A calendar is a temporal entity defined by using Ladkin's TUS. As TUs are formally represented in a hierarchy of linear intervals, recurrent activities are described by *non-convex* intervals as suggested in [Ladkin 86a]. Granularity is obtained by decomposing all TUS into contiguous partially ordered sequences of other TUs. However, there is no mechanism to obtain inferences about processes working at different time scales and to representing interaction between processes at different grains of time.

This work and those presented above, treat time as a mathematical structure based on natural numbers, or integers, or reals, etc.Almost no attempt has been made to develop a temporal logic theory based on "natural" time. Such natural notions of time include day, lunar month, and seasonal cycles. Furthermore, these theories do not include cyclical aspects of time in their models. Such a need has also been addressed in [Pachet *et al.* 95] for dealing with musical objects.

A framework is needed that allows us to reason about both aspects (cyclicity and granularity). While a linear structure is suitable for granularity, it does not resemble cyclicity. All of these theories try to impose a cyclical interpretation on a linear view of time. I prefer to take a different approach by defining a chain of modular sets (called modular temporal classes) and thus obtain both concepts at the same time. The first demonstration of this idea was proposed in [Mota 94], where I investigated the use of this theory to represent the temporal knowledge of a restricted class of ecosystems. In Chapter 3 I shall present a more detailed description of this theory.

2.6 Summary

We can summarise the discussion on this chapter as follows.

- The use of simulation clauses for representing simulation models of ecosystems has two advantages over traditional computational logic.

    1. temporal knowledge is separated from knowledge about the state of the

world, indexed by the temporal expression.

2. specifications in simulation clauses can be executed, so that we may construct a model to predict the state of the world in future based on the present.

- the main challenges related to the granularity of time and the integration of different models (at different scales of time) we have to face are

  i - how should the value of an agent's attribute at a time between two consecutive time steps at the agent's scale of time be computed?

  ii - how does an agent's attribute progress during a certain period of time? (for aligned and non-aligned periods at different scales of time in relation to (updating) of the agent's attributes).

  iii- how are *i* and *ii* affected by the composition of different agents to form complex agents, so that we may obtain consistent integration of them?

- A basic simulation clause schema is not enough to represent interactions in a dynamic environment, where models of ecosystems may be moved in and out, although it can solve the challenges addressed above.

- A more general simulation clause schema is necessary to express the "adaptability" of a model with respect to changes in the environment to which it belongs. This brings conceptual problems, and in most cases computation is still problematic.

Chapter 3

A Logic for Time from Natural Phenomena

What, then, is time? I know well enough what it is, provided that nobody asks me; but if I am asked what it is and try to explain, I am baffled.
Saint Augustine (The Confession, Book XI, section 14).

3.1 Introduction

Most of the mathematical frameworks for representing cycles we saw in Section 2.5 are based on universal quantification over temporal variables. Although this is a simple method for representing cycles, there is no syntactical representation of the cycles, themselves. The problem is that it is not easy for users to see cyclicity by means of an interpretation of a potentially infinite set.

As we saw in Chapter 2, there is no temporal (logic) representation and reasoning framework which deals with both the granular and cyclical aspects of time. This thesis presents a particular framework for accommodating granularity and cyclicity within a discrete temporal representation, despite the possibility that the ecological processes being represented may be continuous. Approximating continuous change using discrete models is commonplace in the ecological modelling community so this choice is not controversial. However, when combined with notions of cyclicity and granularity, it raises some interesting philosophical issues which are some of the aims of this chapter.

In this chapter we shall discuss these questions by presenting a mathematical structure of time which embodies both cyclicity and granularity. Its basic concepts were first

proposed in [Mota 94], where I investigated the representation of temporal knowledge of ecosystems, mainly the aspects related to ecological seasons. The logic I called **NatureTime** was further proposed as a suitable language for specifying simulation models of ecosystem [Mota *et al.* 95], and we will see it later on in this thesis.

3.2 Time: Change, Cyclicity and Granularity

> All things from eternity are of like forms and come round in a circle.
>
> *Marcus Aurelius Antonius.*

The cyclic aspect of time has been present in philosophy since Plato who proposed a cosmic cyclicity based on the motion of the sun[1]. As time is intimately related to our observations of changes in the physical world, the various religions' concepts about the creation of the universe also exerted a great influence in the advent of cosmological theories to explain time. In contrast to the cyclical view, the idea of a linear time originated in Judaism and was later adopted by other religious doctrines of Europe. Such a view is based on the belief that time is a definite sequence of events, or a history set [Newton-Smith 80].

In both views, the changes in the world can be observed at different grains of time, as Aristotle had already proposed that change does occur at many (possibly infinite) different sizes of time interval [2]. Because the proposals for cyclical time were a pure closed-time structure (i.e. time moments are isomorphic to points on a circle), causal loops made such an idea incoherent. The problem is that the nature of directionality in closed-time worlds does not exist as [Poidevin 95] suggests. As a consequence, the most investigated mathematical models of time concentrated only on changes at different grains of linear temporal intervals and left cyclicity out. Figure 3.1 represents the three main influences on our concept of time, where dashed arrows indicate the lack of a theory of joint change in a cyclical way and at many granularities, and solid double arrow the combination of both ending points in the same framework of time.

These two ancient ideas found scientific basis to support them, but no one has been

[1] Plato - Complete Works, Def. 411a, 411b, 411c, Hackett Publishing Company, Inc., 1997.

[2] Aristotle, *Physics*, VI, esp. 232^b20 ff

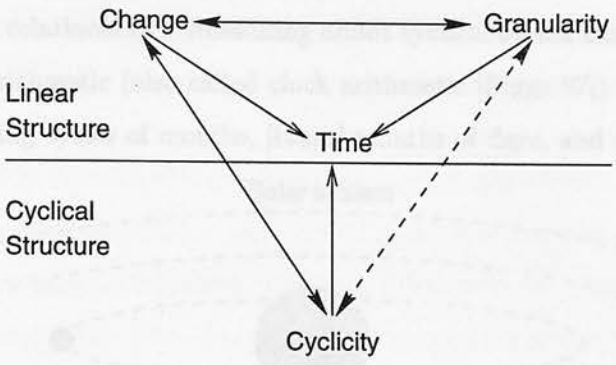

Figure 3.1: Time is a trinity of change, cyclicity and granularity. Cyclicity has not been combined to other two concepts.

concerned with a possible alternative model to cope with them together. While the discovery of the "almost" cyclical paths described by the planets and stars sustained the cyclical view of time, the second law of thermodynamics gave strong support to the idea of strict linear flow of time. The reason is because there seems to exist evidence that there was a starting time and an ending one.

Two other branches from these perspectives evolved. The first is that time progresses despite the apparent degeneration of the universe. In the second the macro cosmos is in a steady-state universe but there may exist local degenerative processes [Davies 95], i.e. time starts and stops for each system but they are not related at all. This same idea had been already presented in [Kardec 68], where time is seen just as a sequence of events related to each celestial system. The next issue is whether time is *bounded* or *infinite*. For practical reasons the time observed for ecological systems is a finite one. Here, I assume that (at least for simulation models of ecosystems) there is a start (which may be independent for each one) and no end in terms of the overall environment.

The sort of common sense knowledge I use to propose a framework of time which embodies both cyclicity and granularity are: the weather changes in a cyclical way, as does the movement of the moon which affects the seas and oceans, and most noticeably our daily experience of the sunlight disappearing and appearing again, the recurring movement at the macro and micro cosmos level (at least as a model). Figure 3.2 shows the natural phenomena which "suggest" that granularity of time is actually a hierarchy of cycles: mathematically speaking, modular sets which are related by a

kind of "inclusion relationship". Reasoning about cyclical events can be done by using simple modular arithmetic (also called clock arithmetic [Biggs 87]). In this way years can be seen as being cycles of months, [lunar] months of days, and so on.

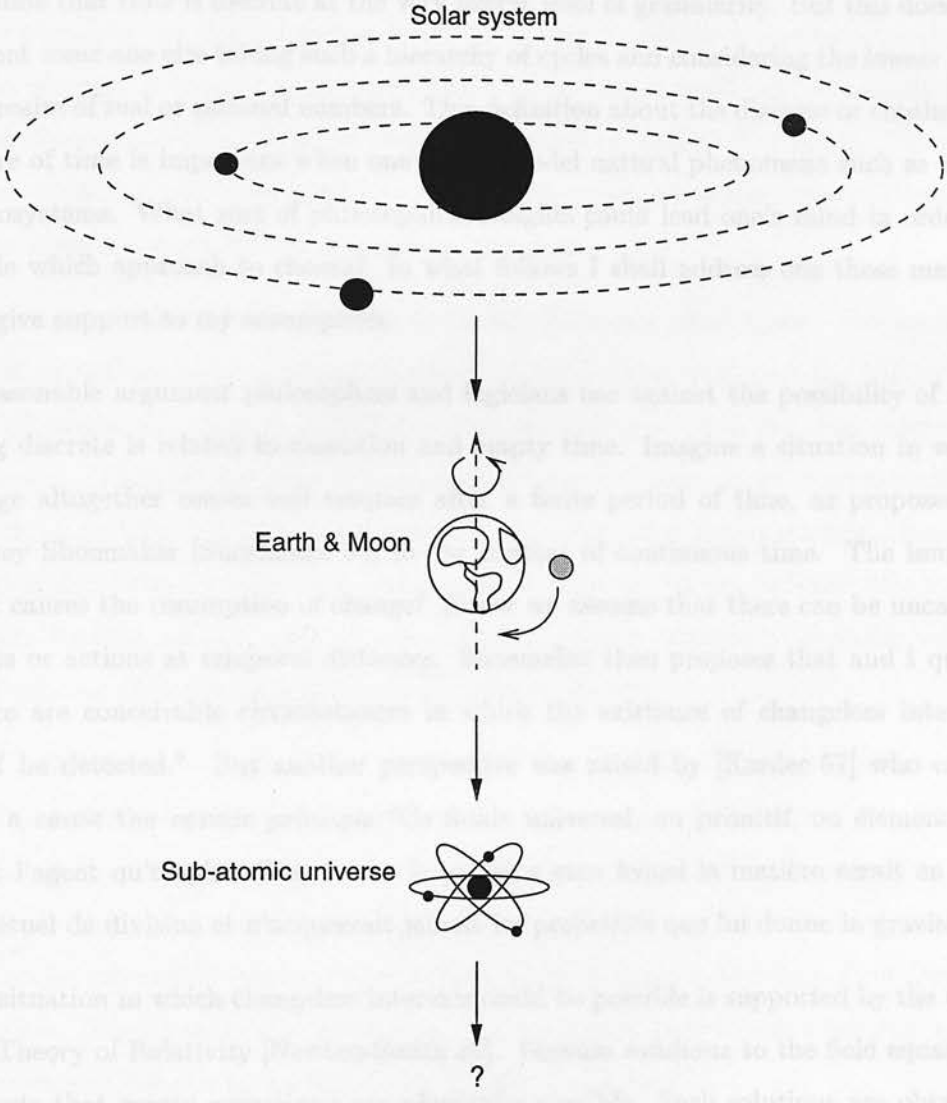

Figure 3.2: View of the natural events commonly used to refer about time. Years are cycles of months, months of days, and so on.

Within this model we assume, also, at all levels, except the highest being considered, time is closed, i.e. isomorphic to points on a circle. Note that if we consider all levels as circular we end up with the pure cyclical view of the universe; if we consider time as finite then such a hierarchy can be mapped to the linear structure. In this respect this framework generalises all others. The final consideration is whether time is dense of discrete.

3.3 Discrete, Continuous or Dense? "Illusion" of the Same Thing

I assume that time is discrete at the very lowest level of granularity. But this does not prevent some one else taking such a hierarchy of cycles and considering the lowest level as a realm of real or rational numbers. This definition about the discrete or continuous nature of time is important when one has to model natural phenomena such as those of ecosystems. What sort of philosophical insights could lead one's mind in order to decide which approach to choose? In what follows I shall address one these matters and give support to my assumption.

A reasonable argument philosophers and logicians use against the possibility of time being discrete is related to causation and empty time. Imagine a situation in which change altogether ceases and resumes after a finite period of time, as proposed by Sydney Shoemaker [Shoemaker 95] in the context of continuous time. The issue is, what causes the resumption of change? Either we assume that there can be uncaused events or actions at temporal distances. Shoemaker then proposes that and I quote, "there are conceivable circumstances in which the existence of changeless intervals *could* be detected." But another perspective was raised by [Kardec 57] who called such a cause the *cosmic principle*:"Ce fluide universel, ou primitif, ou élémentaire, étant l'agent qu'emploie l'esprit, est le principe sans lequel la matière serait en état perpétuel de division et n'acquerrait jamais les propriétés que lui donne la gravité".

The situation in which changeless intervals could be possible is supported by the General Theory of Relativity [Newton-Smith 80]. *Vacuum solutions* to the field equations suggests that empty spacetimes are physically possible. Such solutions are obtained if one assumes that the universe represented by the model has no matter or radiation whatsoever. However, Newton-Smith emphasize that although this is a respectable theory, there seems to be some problematic situations with the field equations and some physicists have been trying to provide theories to oppose Relativity theory admission of vacuum solutions. This, he argues, should be enough to block any abrupt conclusion of the illogical possibility of continuity of time. I agree with that mainly because, in fact, even at the subatomic level there is no proof that change is either

discrete or continuous.

Research in Quantum Physics has been trying to find a Quantum Theory of Gravitation [Feyman 92] in order to join *quantum*, gravitational and electro-magnetic forces in one single principle. At the subatomic level, "whenever a particle is confined to a small region of space it reacts to this confinement by moving around, and the smaller the region of confinement is, the faster the particle moves around it" [Capra 92]. The space of this movement can be mapped into a discrete modular set as the limits of Einstein's theory suggests in the following argument that I propose.

Suppose a particle could be confined to an infinitesimally small space. The tendency of it is to achieve higher and higher speed. Since nothing can cross the barrier [3] of light [Einstein 23], there is a limit to which we may subdivide the time interval of movement. Thus, either there is a changeless period of time or we cannot confine a particle to an infinitesimally small space. It is at this point that a particle cannot be detected, but behaves either in a discrete way in form of "energy packets" (discovered by Max Planck), or as a wave of probability. Therefore, physically speaking nobody can really say that change is either continuous or discrete.

Why is this discussion relevant to the topic of this research and to our framework of time? It is well known that when developing simulation models of continuous processes one has to impose discrete limitations on continuous differential equations. This does not imply though that actual time is discrete but that even physicists come across a limit at the sub-atomic level mentioned above when trying to model "actual continuity" of change. It is impractical to go down to the sub-atomic level to represent changes on ecosystems. They can be modelled in a discrete hierarchy.

One might argue that such impositions can be seen as implementations (usually in form of algorithms) of the philosophical assumption that there exists some (yet undetected) cause for the resumption of change. But we do not assume this. The issue is how to relate scale of time with the event associated with such a scale. The modeller should then make a choice of the level of "accuracy" she/he wants of the results by specifying the levels of time scale related to the processes involved.

[3] Einstein's theory does not deny particles travelling faster than light, but in this case they never go slower as Capra (*op. cit.*) explains.

This argument does not deny our perceptions of the continuity of actual change, but if one wants to assume such a cause for the resumption of change it always has to be placed at the lowest level of granularity. The difference to continuous models is that they consider change taking more (potentially infinite) levels of granularity into account. This means that the algorithms for discrete models should not be rigid but flexible enough to allow the introduction of another level of accuracy in the case other models have to be integrated.

3.4 Pragmatic Accounts for NatureTime

After this philosophical exercise I now return to the main purpose of this work: how can we represent change at many scales, and in a cyclically way in computer programs? Taking these considerations above into account, we need to impose some restrictions since computers are finite and discrete machines. Thus, in practice, we have the following aspects of a *Linear-Cyclic Hierarchy* of time.

1. A Temporal class is defined as modular set of another temporal class, so forming a hierarchy of *modular temporal classes* (MTC). This allows us to have multiply nested levels of time granularity.

2. *temporal entity* (TE) - is a reference to a moment of time or a collection of moments. A collection can be a cyclical or a linear interval. For example, June, 24th, year 1980 or all Sundays in 1997 are examples of temporal entities.

3. The number of levels of a time hierarchy should be finite. The highest can be considered as the largest interval, and the lowest as the smallest. This is to make the theory a tractable one.

4. The highest level of the hierarchy is not circularly grouped to form a MTC, but it is linearly ordered in a (potentially) infinite sequence. Thus, for each instance of the highest temporal class there is one positive integer.

The picture we can draw from this perspective of time is a sequence of cycles as depicted in Figure 3.3. The inner circles represent the lowest levels of granularity.

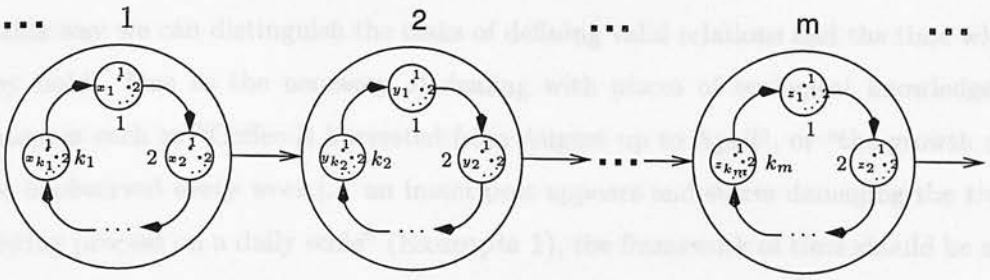

Figure 3.3: The *Linear-Cyclic* structure of time.

The rest of this chapter is dedicated to the technical details of the **NatureTime** logic. Finally I shall present the expressiveness of the language, and briefly discuss its limitations for specifying and executing simulation models of ecosystems.

3.5 NatureTime Syntax

The main idea of the logic is to separate the task of defining relations within the model from the task of computing when such relations hold. In other words, **NatureTime** logic is a quantified temporal logic because there are quantifiers ranging over moments and intervals, where terms stand for temporal relations [Rescher & Urquhart 71]. This does not mean that I do not accept tensed facts and modality, but that it does not suit simulation models of ecosystems (the domain of application being considered in this work). The language itself is Prolog-like with restricted forms of unification on temporal labels. Negation is allowed by failure under the closed world assumption because this language is not for programming open agent-based systems yet. This kind of systems will be treated on Chapters 5 and 6.

Standard computational logic simply introduces an extra argument to each temporally sensitive predicate which denotes the time at which it holds (see [Robertson *et al.* 91] for examples). One problem with this is that nothing can be said about the temporal aspects of assertions [Shoham 88b]. Moreover, when dealing with hierarchical and non-aligned processes a special treatment is necessary of the temporal components of predicates. Thus it is helpful to separate them from the clauses which they label, and so the introduction of operators to associate formulae to temporal terms is more convenient.

In this way we can distinguish the tasks of defining valid relations and the time which they hold. Due to the necessity of dealing with pieces of ecological knowledge in sentences such as "Coffee is harvested from August up to April", or "the growth of a tree is observed every week,[..] an insect pest appears and starts damaging the tree's growing process on a daily scale" (**Example 1**), the framework of time should be able to offer mechanisms for representing cyclical processes at different scales of time. In what follows we describe the syntax of **NatureTime** and its expressive power and limitations. A formal semantics for **NatureTime** is provided in Appendix A, and a meta-interpreter will be presented in Chapter 4.

3.5.1 Vocabulary

The vocabulary is formed by sets of symbols for variables, $\mathcal{L}_v$, constants, $\mathcal{L}_c$, non-temporal functions $\mathcal{L}_f$ and predicates $\mathcal{L}_p$ as in Prolog. This is extended with a countably infinite set of temporal variables $\mathcal{L}_{tv}$, where s_i, t_i, u_i are variables of $\mathcal{L}_{tv}$; a finite set $\mathcal{L}_{tc}$ of temporal constants defined as $\{lowest,\ flow\_time,\ infinite,\ smallest\} \cup \mathcal{T}_C$, where $\mathcal{T}_C$ is a set of names of temporal classes; a finite set $\mathcal{L}_{tf}$ of temporal function symbols f_t/n, where n is the arity of the function, an important subset of this is $\{.../2, p/2,\ i/2,\ t/k,\ of/2,\ before/2,\ after/2,\ plus/2\}$, the last three are written using infix notation; a finite set $\mathcal{L}_p$ of predicate symbols p/n, where $n > 0$ is the arity of p, and special predicate symbols $mod\_temp\_class/3$, $future/3$, $precedes/2$, $down\_equivalent/2$, $equivalents/2$, $mod\_decomposed/3$, $on/2$; the logical connectives for negation, conjunction, disjunction, and implication represented here by $\neg$, &, $\vee$, and $\Longleftarrow$, respectively, where truth value for true is represented by $\top$, and false by $\bot$; the temporal symbol @ (which connect terms to their temporal labels); finally the temporal interval demarcator $\lfloor\ \rfloor$.

3.5.2 Classes of Expressions

The definitions for logical term and atomic formula (AF) are the usual classical notions. The first kind of special expression we need is the definition for modular temporal classes. This is formed by using the special predicate $mod\_temp\_class/3$ as follows, where $\mathbb{Z}_m$ represents a modular set with m elements from the integers starting from 1

up to m.

Definition 3.1 (Modular Temporal Class) *Let c_i and c_j be constants. If we associate temporal units to c_i and c_j so that each unit of c_i contains m units of c_j (or a modular set $\mathbb{Z}_m$), then we can say that c_i is a* modular temporal class *(MTC) defined by a modular set of c_j, written $mod\_temp\_class(c_i, c_j, m)$.*

An important sequence of sentences defining MTCs have the form as follows, where $x...y$, $x, y \in \mathbb{Z}$ is a notation to denote a range between these numbers.

- $mod\_temp\_class(flow\_time, c_n, infinite)$, where $c_n \in \mathcal{T}_C$
- $mod\_temp\_class(c_i, c_{i-1}, m_v)$, where $1 < i \leq n$, $c_i, c_{i-1} \in \mathcal{T}_C$, and $m_v \in \mathbb{Z}^+$.
- $mod\_temp\_class(c_1, lowest, smallest)$, where $c_1 \in \mathcal{T}_C$.

If c_{i-1} defines c_i with modular value m_v, then we say c_i is an *immediate descendant* of c_j. I call any valid sequence of sentences defining MTCs plus the temporal unification (which we shall see in Section 4.5), as a Modular Theory of Time (MoTT). In a MoTT there will always be a unique sequence of MTC definitions from (the lowest) c_1 to (the highest level) c_k called the *Main Time Hierarchy* (MTH). When there exist other MTCs which do not belong to the chain of the MTH, then I call this an extension of a MoTT, or MoTT augmented (MoTTa for the sake of simplicity). For instance, suppose we choose $\mathcal{T}_C = \{day, month, year\}$, then we may assert.

$mod\_temp\_class(flow\_time, year, infinite)$.
$mod\_temp\_class(year, month, 12)$.
$mod\_temp\_class(month, day, 30)$.
$mod\_temp\_class(day, lowest, smallest)$.

These relations define a MoTT where the flow of time, represented by $flow\_time$, is a linear and infinite ($infinite$) sequence of *years*, *year* is a MTC of 12 *months*, *month* is a MTC of 30 *days*, *day* is the smallest (*smallest*) time interval. Note that months are regarded as regular MTC. Although the logic allows us to define real calendars

with non-regular MTC (if $m_v = x \ldots y$ and $x, y \in \mathbb{Z}^+$ and $x < y$), I will omit the details in this thesis because they are less important when considering models of ecosystems. An example of how this might be done can be found in [Mota *et al.* 96].

One may extend this hierarchy to obtain a MoTTa by adding more assertions of MTCs which are not in the MTH. This work will only make use of this to allow users to define different scales (or lengths) of time, but we will not describe intervals on this scales. For example, the following sequence of assertions extend MoTT above.

$mod\_temp\_class(fortnight, week, 2)$.
$mod\_temp\_class(week, day, 7)$.

Note that, in cases like this, one must not define another highest level of time. This means that there will always exist only one highest level being assumed, although one may define a MTC with a modular value higher than the one for the MTC representing the flow of time. In this example, $\mathcal{T}_C$ was extended with $\{fortnight, week\}$. Now, I shall define some valid classes of expressions, where the capital letters A, B, C are used for formulae.

- a *temporal term* (TT) is a temporal variable $s \in \mathcal{L}_{tv}$, or a temporal constant $c_t \in \mathcal{T}_C$, or a term of the form $f_t(\vec{args})$ and $f_t \in \mathcal{L}_{ft}$ in one of the following forms.
  - a *period*, is recursively defined as a 1) *single period* $p(s, m)$ where $s \in \mathbb{Z}^+$ and $m \in \mathcal{T}_C$; or 2) a *composite period* P *plus* P', where P is *single period*, and P' is a *period* term. Let $\mathcal{P}$ be the set of elements of this form.
  - a *smallest temporal entity* (STE) $t(x_1, \ldots, x_k)$, where for n as the number of elements in $\mathcal{T}_C$, then $k \leq n$, each $x_i \in \mathbb{Z}^+$, and each i correspond to exactly one element of $\mathcal{T}_C$. Let $\mathcal{T}_M$ be the set of elements of this form (moments of time).
  - a *linear interval* $s_1 ... s_2$, where s_1 and s_2 are STEs. Let $\mathcal{I}_L$ be the set of elements of this form.
  - a *collection interval*

* $f_t(n)$, where $f_t \in \mathcal{L}_{tf}$, and $n \in \mathbb{Z}^+$, and f_t is some $\alpha \in \mathcal{T}_C$. For instance, the function symbol of $week(1)$ corresponds to first *week* of $\mathcal{T}_C$
* $I\ of\ S$, where I is a cyclical interval and and S is a *collection interval*.

Let $\mathcal{C}_I$ be the set of elements of this form.

– a *cyclical interval* $i(s_1...s_2, c)$, where $c \in \mathcal{T}_C$, and either

* $s_1, s_2 \in \mathbb{Z}_m$
* $s_1 = \alpha(x_1)$ and $s_2 = \alpha(x_2)$, where $\alpha \in \mathcal{T}_C$ and $x_1, x_2 \in \mathbb{Z}_m$ such that $mod\_temp\_class(c, \alpha, m)$ holds.

Let $\mathcal{I}_\circ$ be the set of elements of this form.

Examples of TTs are $p(1, day)$, $p(5, day)\ plus\ p(1, week)$, $t(17, 7, 1994)$, $i(9...2, month)$, $i(day(5)...day(1), week)$, and $i(day(5)...day(1), week)\ of\ year(1996)$.

- a pure temporal expression (PTE) is the set $\mathcal{P}_{TE} = \mathcal{T}_M \cup \mathcal{I}_L \cup \mathcal{C}_I \cup \mathcal{I}_\circ \cup \{t \mid t$ is in the form $p\ after\ t'$, where p is a *period* term, and and $t' \in \mathcal{P}_{TE}\}$. An example of a PTE is $p(24, year)\ after\ t(17, 7, 1970)$.
- a non-explicit temporal moment is $\lfloor T \rfloor$ if either $T \in \mathcal{T}_M$ or it is in the form $p\ after\ T'$ and $T' \in \mathcal{T}_M$. This kind of notation is useful to delimit intervals where the right ending point is not included but the immediate moment before it. For example, $attack(bug\_pest, tree)$ @ $T...\lfloor p(1, week)\ after\ T \rfloor$.
- *atomic temporal formula* (ATF) is either an AF or an AF annotated with a PTE by using the temporal operator "@", i.e. if A is an AF and T is a PTE, then A @ T is an ATF.
- *body* is in one of the forms $A\ \&\ B$, $A \vee B$, C or $\neg C$, where A and B are bodies and C an ATF.
- a well formed temporal formula (WFTF) is either an ATF A with body equal to $\top$, or $A \Longleftarrow B$ where A is an ATF and B is a body. A is called the *head*. A WFTF whose its *head* is an AF, and whose *body* is formed only by AFs is a Prolog clause with no temporal contents. An example of a WFTF is the part of the specification of the attack of the *bug_pest* to the leaves of the *tree* in **Example 1.**

$attack(bug\_pest, tree)$ @ $p(1, month)$ $after$ T

$\Longleftarrow$

$eating(bug\_pest, leaves(tree))$ @ T &

$attack(p_j, bug\_pest)$ @ T &

$resisted(bug\_pest, p_j)$ @ $T...\lfloor p(1, month)$ $after$ $T\rfloor$.

This means *the bug_pest attacks tree 1 month after T if it was eating the leaves of the tree at T and it was attacked by pesticide p_j at T and it resisted the effects of p_j since T until the moment before the current attack.*

3.6 Expressiveness and Limitations

Now, we shall see some examples of how to use the language. I shall use the **Example 1** presented in Section 1.1 and represent some of its information: "*John* planted a *tree*, and started observing its growth every week. After some time an insect pest appears and starts damaging the *tree*'s growth in a daily scale. *John* decides to use some chemical resource p_j, from *Johnston* chemical producers, against the *bug_demon* which goes away, but some portion of p_j stays on the leaves of the *tree* which absorbs it in a few hours. The *bug_demon* returns after some time and more chemical needs to be sprayed. This occurs at the same time every month." In what follows the MTC declarations are

$mod\_temp\_class(flow\_time, year, infinite)$.
$mod\_temp\_class(year, month, 12)$.
$mod\_temp\_class(month, day, 30)$.
$mod\_temp\_class(week, day, 7)$.
$mod\_temp\_class(day, lowest, smallest)$.

Example 5 I chose $t(1, 1, 1)$ as starting time (1st day of the 1st month of the 1st year). The fact that *John* observes the growth of the *tree* can be understood as *John* measures the *tree*'s height every week. Assuming the initial *tree*'s height is 1m; the insect pest appears at $t(20, 2, 1)$ and it eats 0.5 days later. We may have, for example, the following representation.

$plant(john, tree)$ @ $t(1,1,1)$.

$observe(john, height(H), tree)$ @ $T \Longleftarrow value(height, tree, H)$ @ T.

$value(height, tree, 1)$ @ $t(1,1,1)$.

$value(height, tree, H)$ @ $p(1, week)$ $after$ $T_p \Longleftarrow$

$value(height, tree, H_p)$ @ T_p &

$influence(bug\_pest, I)$ @ $T_p ... \lfloor p(1, week)$ $after$ $T_p \rfloor$ &

$resource\_acquired(tree, soil, R)$ @ $T_p ... \lfloor p(1, week)$ $after$ $T_p \rfloor$ &

$\mathcal{G}(H_p, I, R, H)$.

$attack(bug\_pest, leaves(tree, X))$ @ $t(20,2,1) \Longleftarrow$

$leaves(Y, tree)$ @ $t(20,2,1)$ &

X is $0.5 \times Y$.

$attack(bug\_pest, leaves(tree, X))$ @ $p(1, day)$ $after$ $T_p \Longleftarrow$

$attack(bug\_pest, leaves(tree, \_))$ @ T_p &

$\neg attack(\_, bug\_pest, pesticide(X))$ @ T_p &

$leaves(Y, tree)$ @ $\lfloor p(1, day)$ $after$ $T_p \rfloor$ &

X is $0.5 \times Y$.

$attack(bug\_pest, leaves(tree, X))$ @ $p(10, day)$ $after$ $T_p \Longleftarrow$

$attack(bug\_pest, leaves(tree, \_))$ @ T_p &

$attack(\_, bug\_pest, pesticide(X))$ @ T_p &

$leaves(Y, tree)$ @ $\lfloor p(10, day)$ $after$ $T_p \rfloor$ &

X is $0.5 \times Y$.

Note how useful the temporal interval demarcator is for representing the period of influence and resources acquired from the soil by *tree*. The delimiter does not mean the influence or that the resource acquisition stops at the new time stamp, just that to compute the new value for a certain time t only those influences which happened before should be taken into account. To complete the specification I assume that the *bug_peste* re-appears at the last period starting on Tuesday until Sunday of every month.

Example 6

$attack(john, bug\_pest, pesticide(p_j))$ @ $T \Longleftarrow$

$$attack(bug\_pest, leaves(tree, \_)) \ @ \ T \ \&$$
$$has(john, pestecide(p_j)) \ @ \ T.$$
$$attack(bug\_pest, leaves(tree, X)) \ @ \ last(i(day(3)...day(1), week) \ of \ i(X...X, month))$$
$$\Longleftarrow$$
$$attack(\_, bug\_pest, pesticide(X)) \ @ \ p(1, month) \ before \ last(i(day(3)...day(1), week) \ of \ i(X...X, month)).$$

A careful observation shows that we have made many temporal assumptions which in practice we cannot justify. For example, nobody really knows when a plague of insects will attack a plantation nor for how long will it be free from such a plague. Thus, the language does not provide temporal connectives for uncertain temporal knowledge (for this I mean temporal operators like $\Diamond$, etc where no explicit representation of time duration and quantification over temporal variables are necessary). Most of traditional temporal logics (for example [Gabbay 87, Gabbay 89, Barringer *et al.* 89]) are based on this approach and an interesting extension of **NatureTime** is to mix both kind of representations. However, this was not the purpose of this work and assumptions such as those above are quite enough for the domain of simulation models of ecosystems.

Simulation models of ecosystems are only useful if they can be executed. In [Mota 94], I show how inefficient the execution of simulation models can be when the flow of time is represented by recursion over time points. This has been already pointed out in [Robertson *et al.* 91] for standard computational logic. Moreover, the main problem is that the value of things cannot be obtained for time stamps in between two consecutive time steps, i.e. reasoning about processes non-aligned in time is not possible. The next chapter presents a temporal substitutional framework for dealing with temporal unification, and shows how **NatureTime** can be efficiently implemented, so that a considerable class of simulation models can be represented and executed.

3.7 Summary

- From the three aspects related to time, change, granularity and cyclicity, only the first two have been investigated and mathematical frameworks proposed to represent them. There was no alternative approach to treat them avoiding the problems that traditional formalisations have found.
- The philosophical account for time is that it is a linearly ordered hierarchy of cyclical intervals. In such a hierarchy, discrete models of time are possible by imposing a limit in the number of levels of time hierarchy. As far as simulation models are concerned this assumption is enough to represent changes at many scales.
- The closest mathematical framework for expressing this concept is a hierarchy of modular sets. **NatureTime** logic is based on such a framework and it is expressive enough to represent temporal cycles and change at various scales of time.
- In order to use **NatureTime** to representing simulation models we need a more sophisticated meta-interpreter to cope with temporal unification and re-use computed goals during the search. To this search, a little of ingenuity must be added in order to offer inference mechanisms to reason about processes non-aligned in time.
- The logic we saw in this chapter does not provide mechanisms for representing uncertain temporal knowledge where the use of operators such as "sometime" are necessary.

Chapter 4

Temporal Unification and Meta-Interpreter

4.1 Introduction

NatureTime logic was presented as a language for representing interaction of agents and simulation models of ecosystems running at different scales of time [Mota *et al.* 96, Mota & Robertson 96]. In the last chapter we saw its philosophical background and the mathematical structure of the language. We saw how expressive the language can be for representing a certain class temporal knowledge, and also for specifying simulation models. However, we did not see how temporal unification works during temporal deduction.

Temporal unification handles labels and terms related to the temporal universe in a special way. From the point of view of classical logic, temporal labels and terms are quantified (not necessarily ground). In this case, the deductive system operates on constants and function terms. Instead of having to test for equality between such expressions I use a test for unifiability between them. This was originally suggested in [Frisch 91], where a quantified term is interpreted as a schema for the set of its ground instances. In that work, Frisch uses this idea to propose a *substitutional framework* for integrating logical deduction and sortal deduction to form a deductive system for sorted logic. In [Frisch & Page 95], this approach is extended to allow a *constraint theory* as background information rather than a simple sort theory.

The approach I propose here is not a sorted one because there is no need to define

sorts. Sort theory provides mechanisms for making general claims about individuals in a certain domain class rather than the elements of the entire universe. I prefer to make a general claim about temporal individuals in a subset of the entire universe and which are used to index possible states of the world. The framework used in this work will be presented in this chapter along with some outlines about the correctness of how to manipulate it to find answer substitutions for temporal queries. I shall also introduce mechanisms for dealing with interacting agents which will be the basis for the multi-agent architecture presented in the next chapter.

4.2 Temporal Variable and Substitutional Framework

Because temporal reasoning in **NatureTime** is seen as a problem of unifying PTEs, it is necessary to control the part of the unification which deals with it. For this I separate a well formed temporal formula from its temporal terms so that temporal unification can be handled apart from the standard unification. Every PTE which annotates an ATF is replaced by a variable (called a temporal variable, *t-variable* for short). Then I treat the temporal term as a binding for that variable. The consequence of this is that substitution has a different treatment when involving *t-variables*. The standard notion of substitution is still used in the non-temporal part of a temporal formula, and as we will see in some components of temporal binding. There is little difference between the usual substitution, as presented in [Lloyd 84], and temporal substitution as the following definition will show.

Definition 4.1 (Temporal Substitutional Framework) *A* Temporal Substitutional Framework *(TSF or* t-substitution*) is a finite set of the form* $\{tv_1/tb_1, \ldots, tv_n/tb_n\}$, *where each* $tv_i \in \mathcal{L}_{tv}$ *is a* t-variable, *each* $tb_i \in \mathcal{P}_{TE}$ *and* $tv_1, \ldots, tv_n$ *are distinct.*

The binding or instantiation of temporal *t-variables* is not made in the usual way, but rather by instantiating the binding associated to it according to the underlying MoTT(a). In order to avoid confusion between "bound" as commonly used in logic programming jargon and a temporal variable being bound to a PTE, I shall call it *t-bound.* A *t-variable* is *t-free* if its corresponding temporal binding is a *t-variable*, and it is *t-bound* or temporally instantiated if its binding is a PTE. We should interpret such

bindings as *schema* for temporal information. This means that standard substitution cannot be applied in *t-substitution.*

Suppose we have the formula A @ T_1 and *t-substitution* $\{(T_1, i(1...4, month))\}$. A classical *substitution instance* for this would be A @ $i(1...4, month)$. If we want to resolve this formula with A @ T_2 & B @ T_2 with *t-substitution* $\{(T_2, i(3...6, month))\}$, then the resolvent formula should be B @ T_1 with *t-substitution* $\{(T_1, i(3...4, month))\}$. However, if we use the original notation for WFTF and apply standard substitution, then either we fail in the attempt of unifying the temporal terms or we have to use an unclear notion of substitution to avoid problems with shared temporal variables, as in [Mota *et al.* 96]. For this reason the application of *t-substitution*, in the usual sense, is delayed until the very end of the deduction and only the bindings will be manipulated during deduction.

During deduction, the correct set of *t-substitutions* is composed by manipulating the bindings of *t-variables* according to the underlying MoTT. Thus we need a standard form of temporal formula so that the substitutions may be consistently generated. For this the restrictions over the range of a *t-variable* and PTEs are now defined.

4.3 Restrictions on T-variables and Temporal Normal Form

The restrictions operate on *t-variables* wherever they occur in a temporal clause. The restrictions we are going to impose are necessary as temporal entities are function terms of classical logic, and so there may be clashes of terms during standard unification. However, there are some levels of the arguments of such terms in which standard unification is necessary. Therefore, there are other restrictions on a temporal entity. In this way the deductive system has two levels of restriction.

When dealing with sentences of the form A @ T, while A can be any classical formula representing an event, a process, an action, etc., T can only range either:

a. over $\mathcal{T}_M \cup \mathcal{I}_L \cup \mathcal{I}_o$, where the restrictions of the components for each case are those presented in Section 3.5.2, or

b. over $\mathcal{P}_{TE}$, i.e. a term in the form P *after* T. The restrictions here are more

complex. The constraints are:

- P ranges over the set of elements of the form $p(D, C)$, called a (simple) period of time, where $D \in \mathbb{Z}^+$ and $C \in \mathcal{T}_C$.
- T ranges over $\mathcal{P}_{TE} \cup \mathcal{T}_M \cup \mathcal{I}_L \cup \mathcal{I}_o$.
- both P and T cannot be *t-variables* at the same time.
  1. If P is a standard variable, then T must be *t-bound*
  2. If T is a free *t-variable* then P must be instantiated to a ground term.

The last two restrictions are useful to obtain which period of time is related to T and the TE represented by $P\ after\ T$, and to know which TE in the past (in this case T) is related to a TE in the future (in this case $P\ after\ T$) and the period between them (i.e. P), respectively. In Table 4.1 summarises the allowed forms of PTEs (regarded in this work) and their corresponding representation as temporal variables in a TSF. In what follows CPTE stands for canonical PTE; X and Y are variables; expressions in the form $i(Z_1...Z_2, C)$, $t(X_1, \ldots, X_n)$; $S...T$ where $S, T \in \mathcal{T}_M$, are CPTEs; a *t-substitution* tv/t_b is represented by (t_v, t_b).

	PTE	Corresponding term in a TSF	Restriction
1	X	$\{(X, Y)\}$	$var(X)$, $var(Y)$
	$i(Z_1...Z_2, C)$	$\{(X, i(Z_1...Z_2, C))\}$	$Z_1, Z_2 \in \mathbb{Z}^+$, $C \in \mathcal{T}_C$ or $var(Z_1)$ and $var(Z_2)$
2	$t(X_1, \ldots, X_n)$	$\{(X, t(X_1, \ldots, X_n))\}$	$X_i \in \mathbb{Z}_{mi}$,$C_i \in \mathcal{T}_C$ or $var(X_i)$ for $i = 1, \ldots, n$
3	$S...T$	$\{(X, S...T)\}$	$S, T \in \mathcal{T}_M$ or $var(S)$ and $var(T)$
4	$P\ after\ T$	a - $\{(X, P\ after\ T)\}$ b - $\{(X, P\ after\ T), (T, Y)\}$	a - T is a CPTE or b - $ground(P)$ and $var(T)$

Table 4.1: Allowed forms of pure temporal expressions and their corresponding normal forms.

Now we can define the standard form we need for temporal formula. For sake of clarity I shall use union of sets when appropriate.

Definition 4.2 (Temporal Normal Form) *Let A be a WFTF and $T_1, ..., T_n$ PTEs occurring in A. $A' :: \theta$ is the* temporal normal form (TNF) *of A, where A' is the*

result of replacing every ATF $A_i \ @ \ T_i$ *in* A *with* $A_i \ @ \ X_i$ *where* X_i *is a new variable,* $\theta = \{(X_1, T_1), \ldots, (X_n, T_n)\}$ *is the t-substitution.*

The computation which transforms a temporal formula into its TNF is given in the following algorithm.

Algorithm 1 Rewrite a Temporal Formula in to a TNF using the following rule set:

Require: A is a WFTF.
Ensure: $A' :: \theta$ is the TNF of A.
Rule 1: A is a classical AF
$A \rightarrow A :: \varnothing$, or $A \rightarrow A \Leftarrow \top :: \varnothing$, in the case A is also an assertion.
Rule 2: A is $A \ @ \ T_1 ... T_2$ and $T_1, T_2 \in \mathcal{T}_M \cup \mathcal{I}_L \cup \mathcal{I}_\circ \cup \mathcal{P}_{TE}$
$A \ @ \ T_1 \rightarrow A \ @ \ X_1 :: \theta_1$
$A \ @ \ T_2 \rightarrow A \ @ \ X_2 :: \theta_2$
$A \ @ \ T_1 ... T_2 \rightarrow A \ @ \ T :: \{(T, X_1 ... X_2)\} \cup \theta_1 \cup \theta_2$
Rule 3: A is $A \ @ \ T$ and T is a standard variable
$A \ @ \ T \rightarrow A \ @ \ T :: \{(T, V)\}$, where V is a new variable for the binding of T.
Rule 4: A is $A \ @ \ T_e$ and T_e is a canonical PTE
$A \ @ \ T_e \rightarrow A \ @ \ T :: \{(T, T_e)\}$, T is a new temporal variable with binding T_e.
Rule 5: A is $A \ @ \ P \ after \ T_e$ and either P is a ground period and T_e a variable or T_e is a canonical PTE or $T_e \in \mathcal{T}_M \cup \mathcal{I}_L \cup \mathcal{I}_\circ$
$A \ @ \ P \ after \ T_e \rightarrow A \ @ \ T :: \{(T, P \ after \ T_e)\}$
Rule 6: A is $A \ @ \ P_1 \ after \ T$ and $T \in \mathcal{P}_{TE}$
$A \ @ \ T \rightarrow A \ @ \ T_2 :: \theta_2$
$A \ @ \ P_1 \ after \ T \rightarrow A \ @ \ T_1 :: \{(T_1, P_1 \ after \ T_2)\} \cup \theta_2$ if $T \in \mathcal{P}_{TE}$
Rule 7: A is $A \ @ \ T_1 :: \theta_1 \ \& \ B \ @ \ T_2 :: \theta_2$ and V is the binding of T_1 and T_2
$A \ @ \ T_1 :: \theta_1 \ \& \ B \ @ \ T_2 :: \theta_2 \rightarrow A \ @ \ T_2 \ \& \ B \ @ \ T_2 :: \theta_1 \setminus \{(T_1, V)\} \cup \theta_2$
Rule 8: A is $A_1 :: \theta_1 \ \& \ A_2 :: \theta_2$
$A_1 :: \theta_1 \ \& \ A_2 :: \theta_2 \rightarrow A_1 \ \& \ A_2 :: \theta_1 \cup \theta_2$.
Rule 9: A is $A_1 :: \theta_1 \vee A_2 :: \theta_2$
$A_1 :: \theta_1 \vee A_2 :: \theta_2 \rightarrow A_1 \vee A_2 :: \theta_1 \cup \theta_2$.
Rule 10: $A :: \theta \Leftarrow B :: \delta$
$A :: \theta \Leftarrow B :: \delta \rightarrow A \Leftarrow B :: \theta \cup \delta$.

In what follows I show the transformation of the WFTF of Section 3.5.2 by using **Algorithm** 1 presented above.

By applying rule 6 to the head of the clause results in the following expression.

$$attack(bug\_pest, tree) \ @ \ T_1 :: \{(T1, p(1, month) \ after \ T)\} \cup \{(T, V)\}$$
$$\Leftarrow$$
$$eating(bug\_pest, leaves(tree)) \ @ \ T \ \&$$
$$attack(p_j, bug\_pest) \ @ \ T \ \&$$

$resisted(bug\_peste, p_j)$ @ $T...p(1, month)$ $after$ T.

By applying rule 3 to the first two ATFs and rule 2 to last ATF of the body of the clause:

$attack(bug\_pest, tree)$ @ $T_1 :: \{(T_1, p(1, month)\ after\ T), (T, V)\}$

$\Longleftarrow$

$eating(bug\_peste, leaves(tree))$ @ $T_2 :: \{(T_2, T)\}$ &

$attack(p_j, bug\_peste)$ @ $T_3 :: \{(T_3, T)\}$ &

$resisted(bug\_peste, p_j)$ @ $T_4 :: \{(T_4, T_5...T_6), (T_6, p(1, month)\ after\ T_5), (T_5, T)\}$.

If we apply rule 8 twice in the body of the clause and then apply step 7 also twice we have.

$attack(bug\_pest, tree)$ @ $T_1 :: \{(T_1, p(1, month)\ after\ T), (T, V)\}$

$\Longleftarrow$

$eating(bug\_peste, leaves(tree))$ @ T_5 &

$attack(p_j, bug\_peste)$ @ T_5 &

$resisted(bug\_peste, p_j)$ @ $T_4 :: \{(T_4, T_5...T_6), (T_6, p(1, month)\ after\ T_5), (T_5, T)\}$.

Note that some names of *t-variables* disappeared because they referred to the same binding due to step 7.

Step 4 - By applying rule 10 to whole clause we end up with the following WFTNF.

$attack(bug\_pest, tree)$ @ T_1

$\Longleftarrow$

$eating(bug\_peste, leaves(tree))$ @ T &

$attack(p_j, bug\_peste)$ @ T &

$resisted(bug\_peste, p_j)$ @ $T_4 :: \{(T_4, T...T_1), (T_1, p(1, month)\ after\ T), (T, V)\}$.

Note that a temporal variable only appears in the binding of another temporal variable in the case of a non CPTE, e.g. $\{(X, p(1, month)\ after\ T), (T, Y)\}$. As we shall see later, expressions of this type are reduced, in some cases, to a canonical form as soon as the temporal variable T is instantiated to some non canonical form.

We now give the definition of a set of temporal clauses in their temporal normal form as follows.

Definition 4.3 (Temporal Normal Logic Program) *A* Temporal Normal Logic Program *(TNLP) $\mathcal{P}$ is a finite set of temporal clauses in their TNF, i.e.* $\{C_1 :: \theta_1, \ldots, C_n :: \theta_n\}$. *I shall write* $\mathcal{P}^\beta$ *as a short notation for a TNLP, where* β *is the family* $\{\theta_1, \ldots, \theta_n\}$.

4.4 Least Form of a PTE

A temporal variable can be *t-bound* to any temporal expression, for instance $P\ after\ T$, and there may be *t-variables* within such expressions which are bound to other terms. Hence we must get their *reduced form* before temporal unification. In [Mota *et al.* 96] we used a restricted form of reduction of PTE which did not take into account a set of temporal bindings. Because such a set is now being used I prefer to use the term *least form* of a PTE in relation to a given TSF, and the reduction of PTE is called when necessary (Appendix B.3.2 shows the definition of *reduce*/2).

The least form of a PTE is the minimal representation of it, i.e. the simplest structured term that it can be reduced. Such a minimal form is called canonical temporal entity (CTE) and it is a member of $\mathcal{I}_\circ \cup \mathcal{I}_M \cup \mathcal{I}_L$ according to the restrictions of Table 4.1 (rows 1,2 and 3). Note that cyclical or collection intervals do not necessarily have a least form, but rather a minimal set of least forms. Because of such restrictions, the algorithm computes only those canonical forms established in this table. This means that the termination of a least form computation only depends on the length of expressions of the form $P\ after\ P$ (4th row of Table 4.1), which is finite. If the input is not a well formed temporal expression then the computation returns false.

In the following algorithm $var(X)$ is true if X is a standard Prolog variable, $ground(X)$ is true if X is a classical ground term, $canonical\_te(V)$ is true if V is a canonical temporal entity, and $ground\_pte(\theta, T, GT)$ is a computation which returns true if there is a ground PTE instance GT of T is wrt the TSF θ (see Appendix B.3.5 for its specification), and $t\_bind(T, \theta)$ is a function which returns the temporal binding B_T of T in θ assuming $(T, B_T) \in \theta$.

Algorithm 2 Computation of a least Form of a PTE

Require: θ is a TSF and T is a temporal expression.
Ensure: A least form of T is V_f in relation to θ, written $least\_form(\theta, T, V_f)$.
It returns false otherwise.
Case 1: $least\_form(\_, V, V)$
$var(V)$ or $canonical\_te(V)$.
Case 2: $least\_form(\_, P\ after\ T, V_f)$
if $ground(P)$ and $ground(T)$ **then**
$reduce(P\ after\ T, V_f)$
end if
Case 3: $least\_form(\theta, P\ after\ T, V_f)$
if $ground(P)$ and $var(T)$ **then**
$ground\_pte(\theta, T, GT)$
$reduce(P\ after\ GT, V_f)$
end if
Case 4: $least\_form(\theta, P\ after\ T, P\ after\ T_s)$
if $ground(T)$ and $var(P)$ **then**
$T_s = T$
else if $ground(P)$ and $var(T)$ and $\neg\ ground\_pte(\theta, T, \_)$ **then**
T_s is $t\_bind(T, \theta)$
else if $ground(P)$ and $var(T)$ and $\neg\ ground\_pte(\theta, T, \_)$ and
$t\_bind(T, \theta)$ is a non instantiated variable **then**
$T_s = T$
end if

For instance, if $\theta = \{(T, p(1, month)\ after\ X), (X, t(1,1,1))\}$, then the least form of $p(1, month)\ after\ X$ is $t(1,2,1)$. If X was bound to a variable the expression would not change.

The correctness of **Algorithm** 2 is based on two things. First is the constraints of Table 4.1 dealt by each case of the algorithm. Second, is the reduction of expression of the form $P\ after\ T$, $reduce(P\ after\ T, V)$. The formal meaning of an expression like $P\ after\ T$ is given in Appendix A.2.2, and it is based on the up-wave modular sum (Appendix B.2). The reduction algorithm (Appendix B.3.2) relies on the computation which relates a moment of time, another moment in the future and the period of time between them.

Such a computation ($future(T_p, P, T_f)$), always terminate because the up-wave modular sum (or subtraction if it is the case) either returns false or the *waving* effect of the sum (or subtraction) stops in a parameter of $t(x_1, \ldots, x_n)$ which corresponds to the highest (or lowest) level of time granularity. As the number of time scale is finite, if

the constraints are obeyed, then the reduction finds a reduced form of the expression otherwise it returns false. Either $P\ after\ T$ is the least form where restriction 4.b is satisfied, or $P\ after\ T$ is reduced to a term of the form $t(x_1, \ldots, x_n)$ because 4.a is satisfied.

4.5 Temporal Combination

Suppose we have $A \Longleftarrow A_1 \wedge \ldots \wedge A_n :: \theta$, an instance of a temporal clause, and we want to solve A_i with an instance of another temporal clause $B \Longleftarrow C :: \sigma$. Assuming A_i is X_i @ T_i and B is X @ T, we know that the corresponding bindings for T_i and T are in θ and σ, respectively. In a classical resolution step we would need to find the most general unifier (MGU) for X_i and X, add C to the rest of subgoals $A_1 \wedge \ldots \wedge A_{i-1} \wedge C \wedge A_{i+1} \ldots \wedge B_n$ and finally apply the MGU to them. The temporal unification between T_i and T, however, needs to take the underlying MoTT into account to perform a *semantic unification* (Appendix D.3) between T_i and T, and apply its result to θ and σ. I propose a method in which the application of the temporal unifier to other temporal variables is performed during the process of constructing such a unifier.

We can understand this special resolution step as similar to the clocked TiSLD-Resolution of Chronolog(CM) [Liu & Orgun 96]. The difference is that a "**NatureTime**-Resolution" needs a more elaborated unification algorithm between temporal terms because these ones are not simply natural numbers. I will not compare both approaches in depth, but the correctness of their logic will be mentioned afterwards.

The idea of unifying temporal terms is as follows. Given two *t-variables* X and Y, with bindings B_x and B_y, from two distinct *TSFs* θ and σ, respectively. A *Temporal Combination* between σ and θ by *temporally unifying* X and Y is a set $(\theta \backslash \{(X, B_x)\}) \cup (\sigma \backslash \{(Y, B_y)\}) \cup \{(Y, V))\}$, such that V is the *temporal unification* (Appendix D.3) between B_x and B_y. The temporal unification is a more elaborated step because of the special forms of B_x and B_y, taking restrictions of Table 4.1 into account.

We should understand temporal unification as a *semantic unification* between special terms of the Herbrand Universe which are in the form of PTE as we defined in Sec-

Algorithm 3 Temporal Combination

Require: Temporal variables T_s and T_n and their respective TSF θ_s and θ_n.

Ensure: If successful, returns a TSF containing a *t-variable* bounded to a temporal term resulted from the temporal unification between the bindings of T_s and T_n, written as $temp\_comb(T_s, \theta_s, T_n, \theta_n, \theta)$. Otherwise it returns false.

Case 1: $temp\_comb(T_s, \theta, T_n, \sigma, \{(T_s, V)\} \cup \theta \cup \sigma')$

B_{T_s} is $t\_bind(T_s, \theta)$

B_{T_n} is $t\_bind(T_n, \sigma)$

σ' is $(\sigma \setminus \{(T_n, B_{T_n})\}) \setminus \{(T_s, B_{T_s})\}$

if $B_{T_s} = B_{T_n}$ and ($var(B_{T_s})$ or $canonical\_te(B_{T_s})$) **then**

V is B_{T_s}

else if $B_{T_s} \neq B_{T_n}$ and $canonical\_te(B_{T_s})$ and $canonical\_te(B_{T_n})$ **then**

$temp\_unify(B_s, B_n, V)$

end if

Case 2: $temp\_comb(T_s, \theta, T_n, \sigma, \delta)$

$t\_bind(T_n, \sigma)$ is of the form $P\ after\ T_2$

B_1 is $t\_bind(T_s, \theta)$

σ' is $(\sigma \setminus \{(T_2, B_2)\}) \setminus \{(T_n, t\_bind(T_n, \sigma))\}$, where $B_2 = t\_bind(T_2, \sigma)$

if $canonical\_te(B_1)$ and $ground(P)$ **then**

$future(B_3, P, B_1)$

$temp\_unify(B_2, B_3, B_4)$

δ is $\{(T_s, B_1), (T_2, B_4)\} \cup \theta \cup \sigma'$

else if $canonical\_te(B_1)$ and $\neg\ ground(P)$ and $canonical\_te(B_2)$ **then**

$future(B_2, P, B_1)$

δ is $\{(T_1, B_1), (T_2, B_2)\} \cup \theta \cup \sigma'$

else if $var(B_1)$ **then**

$least\_form(P\ after\ T_2, \sigma, B_1)$

δ is $\{(T_1, B_1)\} \cup \theta \cup \sigma'$

end if

Case 3: $temp\_comb(T_s, \theta, T_n, \sigma, \delta)$

if $t\_bind(T_s, \theta)$ is of the form $P\ after\ T$ **then**

$temp\_comb(T_n, \sigma, T_s, \theta, \delta)$

end if

tion 3.5.2. Thus, terms which in Prolog unification would not unify, here may refer to the same temporal entities of a given MoTT, and so are equivalent. This is an important idea for the understanding of the correctness of the temporal combination algorithm I present now.

In the **Algorithm** 3, "\" stands for difference between sets, $future(T, P, F)$ relates the time T and a future time F to the unique period of time between them, and $temp\_unify(B_x, B_y, V)$ is true if the temporal unification of temporal terms B_x and B_y is V (see Appendix B.3 for its specification).

For example, suppose we have a TSF $\theta_1 = \{(T, i(9...6, month)), (Z, X)\}$ and another TSF $\theta_2 = \{(S, i(4...7, month))\}$. Then a temporal combination between θ_1 and θ_2 resulting from the temporal unification between the pair associated with T and S is $\{(T, i(4...6, month), (Z, X))\}$. Another possible temporal combination obtained from temporally unifying S and Z is $\{(T, i(9...6, month)), (Z, i(4...7, month))\}$.

Note that this example suggests that a temporal combination does not necessarily find a unique temporal substitution. But this case only happens when involving collection and cyclical intervals because their least form can be a (minimal) set, as stated before. Thus, I will restrict the correctness of this algorithm only to temporal terms not involving collection, cyclical intervals and linear intervals with variables.

The correctness of **Algorithm** 3 is based on the correctness of **Algorithm** 2, and of *temp_unify/* which performs a time matching (Appendix B.3.1). Because of the restrictions of Table 4.1, then the algorithm either finds a temporal term or return false. But we need to express this correctness in terms of the meaning of the temporal terms involved. The formal semantics of a temporal term is defined in Appendix A, and from this I shall use the notation $[\![\alpha]\!]^{\mathcal{H}_{nt}}$ to mean *the denotation of a well-formed expression* α *relative to a* **NatureTime**-*Interpretation* $\mathcal{H}_{nt}$.

Theorem 1 (Correctness of Temporal Combination) *Let* θ, θ_v, θ_s *be TSFs. Suppose* $(T_v, V) \in \theta$, $(T_v, B_v) \in \theta_v$, $(T_s, B_s) \in \theta_s$, *where* V, B_v *and* B_s *satisfy the restrictions of* Table 4.1, *and* $\theta = (\theta_s \backslash \{(T_s, B_s)\}) \cup (\theta_v \backslash \{(T_v, B_v)\}) \cup \{(T_v, V))\}$. *Then, the temporal combination between* T_v *and* T_s *wrt* θ_v *and* θ_s *is true and* **Algorithm** *3 returns* θ *iff* $[\![V]\!]^{\mathcal{H}_{nt}} \subseteq [\![B_s]\!]^{\mathcal{H}_{nt}} \cap [\![B_v]\!]^{\mathcal{H}_{nt}}$. *Also, if* B_v *and* B_s *are PTEs which involve only time moments (2nd and 4th row of the table), then* V *is unique.*

The proof of this theorem is not in the scope of this work, but we can have an intuition of how this can be done if we relate **NatureTime** temporal unification (or combination) with Chronolog(MC) temporal unification. In this approach temporal atoms are not annotated with temporal terms, but they are clocked and the unification takes clocks definition and clock assignment into account. These, as Liu and Orgun suggested, can be seen as procedures attached to Chronolog(MC) program body.

In **NatureTime**, this would be equivalent to split the temporal annotation of temporally sensitive predicates, and attach the temporal reasoning mechanism in to the programs, also, as procedure calls. The basic difference, as I wrote before, is that **NatureTime** uses a more elaborated notation of temporal terms, and so another level of sophistication is necessary for unification. Furthermore, a clocked fixed-time atom in Chronolog(MC) is similar to an ATF in **NatureTime** where the temporal term is a moment of time.

We shall see now the notions of an answer for temporal substitution and correct temporal substitution. The notion of *answer substitution* I use here is the same as presented in [Lloyd 84].

Definition 4.4 (Temporal Answer Substitution) *Let β be an answer substitution, $\mathcal{P}^{\beta}$ be a TNLP and $A :: \theta$ a temporal goal formula in its TNF. Then a* Temporal Answer Substitution *$\mathcal{P}^{\beta} \cup \{A :: \theta\}$ is a t-substitution for t-variables in θ obtained from the temporal combination between θ using the elements of β.*

Definition 4.5 (Correct t-substitution) *Let $\mathcal{P}^{\Gamma}$ be a temporal normal logic program, G be a goal $\Leftarrow A_1 \& \ldots \& A_n :: \theta$, σ and δ be answer substitution and t-substitution for $\mathcal{P}^{\Gamma} \cup \{G\}$, respectively. We say σ and δ are a* correct substitution *and a* correct t-substitution *for $\mathcal{P}^{\Gamma} \cup \{G\}$ if $\forall(A_1 \& \ldots \& A_n)\sigma :: \delta$ is a logical consequence of $\mathcal{P}^{\Gamma}$.*

We shall see now a meta-interpreter which implements a proof procedure to find a correct *t-substitution* for a temporal goal.

4.6 A Meta-Interpreter for NatureTime Logic

In this section we shall see the specification of a meta-interpreter for **NatureTime** logic based on the concepts of temporal normal form and temporal combination presented in the previous subsections.

4.6.1 Meta-interpreter Specification

The meta-interpreter is an extension of the one commonly used by logic programmers and presented by [Sterling & Shapiro 86] and elsewhere. Because I am also interested in problems where the specification of the behaviour of agents through the flow of time is basically a clause where the state of the agent in the past is related to the (present or) future, then I also allow the meta-interpreter to deal efficiently with this kind of clause. This will be presented in Section 4.7.

The main extension is related to a special unification for PTEs. Note that most of the unification will still be done by standard Prolog unification, except when involving temporal variables and their respective bindings, i.e. PTEs. Because of this, the standard *solve*/1 predicate [Sterling & Shapiro 86] is changed to be a triple relation *solve*/3, where the first two arguments are for a temporal formulae in its TNF, i.e. the first is the temporal formula with temporal variables and the second is its corresponding TSF. The third is an answer substitution.

In what follows, $temp\_formula(X)$ is true if X is a wftf, $var(X)$ is true if X is a variable, $body\_of\_simu\_clause(X \ @ \ T)$ is true if the formula $X \ @ \ T$ matches with part of the body of a simulation clause, and $clause(X \Longleftarrow Y :: U_n)$ is true if there is a clause with X as its head and Y as its body. If Y is $\top$ then X is an assertion, and for non temporal formula we assume that U_n is empty.

1 - $solve(\top, \theta, \theta)$.

2 - $solve(\neg X, \theta, \theta) \Longleftarrow \neg \ solve(X, \theta, \theta)$.

3 - $solve(X, \theta_s, \theta_f) \Longleftarrow clause(X \Longleftarrow Y :: [\,]) \ \& \ solve(Y, \theta_s, \theta_f)$.

4 - $solve(X \ @ \ T_s, \theta_s, \theta_f) \Longleftarrow$

$\neg \ body\_of\_simu\_clause(X \ @ \ T_s) \ \&$

$clause(X \ @ \ T_p \Longleftarrow P :: \theta_p) \ \&$

$temp\_comb(T_s, \theta_s, T_p, \theta_p, \theta_n) \ \& \ solve(P, \theta_n, \theta_f)$.

5 - $solve(A \ \& \ B, \theta_s, \theta_f) \Longleftarrow solve(A, \theta_s, \theta_n) \ \& \ solve(B, \theta_n, \theta_f)$.

6 - $solve(A \vee B, \theta_s, \theta_f) \Longleftarrow solve(A, \theta_s, \theta_f) \vee solve(B, \theta_s, \theta_f))$.

The second clause deals with negation of temporal formulae using the standard negation

by failure. The third clause deals with classical implication on non-temporal clauses. The last two clauses are the cases for conjunction and disjunction and so their meanings are as usual. The fourth clause can be seen as a **NatureTime**-Resolution which is similar to clocked TiSLD-Resolution because the resolution step is also sub-divided into two steps: matching the predicate using standard unification and the temporal combination. Its declarative interpretation for clause 4 is as follows.

4 - *X is true throughout T_s in θ_s with a set of temporal answer substitutions θ_f, if X @ T_s is not in the body of a simulation clause, and X @ T_p is the head of a temporal clause with body P and θ_p as its TSF, θ_n is a temporal combination between θ_s, considering t-variable T_s, and θ_p, considering* t-variable *T_p, and θ_f is the temporal substitution found by solving P with TSF θ_n.*

This meta-interpreter shows more clearly than in [Mota *et al.* 96] where the standard and temporal unification are treated. The solution proposed here can be seen as storing operations over PTE and combining those stored with the new one during the process of deduction. This is similar to the idea of *environment* [van Emden 84] used in the implementation of Prolog machines.

I will restrict the correctness of this interpreter only to canonical instances of temporal normal formulae. By this I mean temporal formulae in which every temporal term is *t-bound* to a moment of time. A similar idea to Chronolog(MC). The other cases are more complicated and of few use for the extensions of this interpreter we shall see from Section 4.7.

Theorem 2 (Correctness of NatureTime Meta-Interpreter) *Let $\mathcal{P}^\Gamma$ be a temporal normal logic program, G be a goal $\Leftarrow A_1 \& \ldots \& A_n :: \theta$, σ and δ be answer substitution and t-substitution for $\mathcal{P}^\Gamma \cup \{G\}$, respectively. Then,* **NatureTime** *meta-interpreter will find σ and δ as* correct substitution *and a* correct t-substitution *for $\mathcal{P}^\Gamma \cup \{G\}$ iff $(A_1 \& \ldots \& A_n)\sigma :: \delta$ is a member of all canonical instances of $\mathcal{P}^\Gamma$.*

This correctness is also restricted to the same limits of standard logic programs. Programs like

$die(t_1)$ @ $T \Longleftarrow \neg die(t_2) :: \{(T, V)\}$.

$die(t_2)$ @ $T \Longleftarrow \neg die(t_1) :: \{(T, V)\}$.

are not considered, even if the temporal terms are ground instances of a time moment term. This situation, which can be understood as a deadlock is also not considered in the agent architecture we shall see afterwards. The proof of this theorem, can be done by induction on the length of the derivation steps of the *solve*/3 procedure. In the next section I will show how to extend this meta-interpreter to reason about interacting agents.

4.7 Reasoning about Interacting Agents

In this section I will present mechanisms of inference which are necessary to deal with agents interacting at many grains of time. The main issues to solve, as I addressed in Section 2.4, are the value of an agent's attribute at a given specific time (which may be in between two consecutive time steps), and the changes of such an attribute during a certain period of time. To tackle this I take advantage of the simulation clause schema as defined in Section 2.2.

The challenge is basically to conveniently guide the search to find precisely the values we want, and for the period requested. First we shall treat the cases where only one value is required, and this divided into aligned and non-aligned search. Then I shall present my solution for representing and reasoning about interactions between agents.

4.7.1 Agents Aligned in Time

For this kind of problem I add two clauses as follows, where $var(X)$ is true if X is a variable, $temp\_subst(U, T, T_s)$ the substitution in TSF U of a temporal variable T is T_s.

7 - $solve(value(Att, Ag, V)$ @ $T, \theta_s, \theta_f) \Longleftarrow$

$temp\_subst(\theta_s, T, T_1)$ &

$var(T_1)$ &

$body\_of\_simu\_clause(value(Att, Ag, V)$ @ $T)$ &

$clause(value(Att, Ag, V_i) \ @ \ T_i \Longleftarrow \top :: \theta_n)$ &

$free\_search(value(Att, Ag, V_i) \ @ \ T_i, \theta_n, value(Att, Ag, V) \ @ \ T, \theta_s, \theta_f)$.

the value of the attribute Att of the agent Ag is V throughout T if T_1 *is the temporal substitution of* T *in* U_s, *and* T_1 *is a variable, and* $value(Att, Ag, V) \ @ \ T$ *can be matched with the body of a simulation clause, and there is a clause* $value(Att, Ag, Vi) \ @ \ T_i \Longleftarrow \top$ *with TSF* θ_n, *and the search free of fixed time yields a solution for* $value(Att, Ag, V)$ *at time* T, *with* T *in* θ_s, *and* θ_f *is its TSF.*

The forward chaining is actually implemented in $free\_search/5$ which uses the state computed at the previous time to generate the next state, according to the level of granularity. As the use of new instances of a simulation clause involves temporal combination to generate them, then I shall define it first. For this, $inst\_simul\_cls/5$ associates $X_p \ @ \ T_p$ of the body of a simulation clause and its TSF θ to the head of this new clause instance, the constraints after $X_p \ @ \ T_p$, as well as in the new TSF resulting from this new instance. This is formalised as follows.

$inst\_simu\_cls(X_p \ @ \ T_p, \theta_p, X_j \ @ \ T_j, Prec, \theta_f) \Longleftarrow$

$clause(X_j \ @ \ T_j \Longleftarrow X_p \ @ \ T_n \ \& \ Prec :: \theta_n)$ &

$temp\_comb(\theta_p, T_p, \theta_n, T_n, \theta_f)$.

$free\_search(X \ @ \ T_i, \theta_i, X \ @ \ T, \theta, \theta_f) \Longleftarrow$

$temp\_comb(\theta_i, T_i, \theta, T, \theta_f)$.

$free\_search(X_p \ @ \ T_p, \theta_p, X_r \ @ \ T_r, \theta_s, \theta_f) \Longleftarrow$

$inst\_simu\_cls(X_p \ @ \ T_p, \theta_p, X_f \ @ \ T, Constraints, \theta_s)$ &

$sub\_tsf((T_p, V_p), \theta_s, \theta_{s'})$ &

$temp\_comb((T_f, V), \theta_f, (T_s, V_s), \theta_{s'}, \theta_n)$ &

$solve(Constraints, \theta_n, \theta_e)$ &

$free\_search(X_f \ @ \ T_f, \theta_e, X_r \ @ \ T_r, \theta_s, \theta_f)$.

Note that T is unified with a free *t-variable* because for this class of problems we gain efficiency if we always keep the temporal variable of the head in its least form. Otherwise, for every new time step required the unification would have to recompute the least form of the temporal variable of the head plus the new instantiation of it.

One might ask "why only for this class of problems?" The reason is that when dealing with simulation clauses we are not interested in the original form of a query, but there may cases in other kinds of temporal query where the original form of the query is extremely relevant. For instance, for a query like "in which month may we harvest grass for hay two months after the first harvesting?", if we reduce its equivalent PTE $p(2, month)\ after\ T$ and don't keep the original query, then we lose the initial meaning and we may obtain a wrong answer. For this case, the reduction on temporal query should be just enough to compute the bindings.

4.7.2 Fixed or Non-Aligned Search

Considering the two kinds of question of Section 2.4, the deduction process should try to satisfy a given query relative to an attribute's value at an specific time T. In [Mota & Robertson 96], the policy proposed for such queries is simply to reason forwards from the first value of the attribute and an instance of the simulation clause for the changes of the attribute, until either $T = T_f$ or $precedes(T_f, T)$, where $T_f = P\ after\ T_p$. In the former case T is aligned to the time steps at which the attribute changes. In the latter, as it is non-aligned, the last value is assumed. However, the meta-interpreter used did not combine this policy with cases in which the agent interacts with others. For example, in the case of an agent interacting at a lower level of time granularity, as in **Example 2**, the deduction process should take into account the fact the other agent has already affected the attribute being observed. Furthermore, there may exist other interacting agents, may be at the same level of time, updating their attribute out of phase with the agent's, as in **Example 3**.

What I propose is to align the search so that the last time stamp found, or V_f will always be at the given T of the query, i.e. $T = T_f$ will always hold. For this, first the solving process needs to "know" the time T_i at which the attribute has its initial value and the period P_a of time between T_i and T. Why is this policy better than the previous one? Suppose one wants to know the value of an attribute of an agent A_1 for time T which is in between T_i and T_k, and $precedes(T_i, T_k)$, which is aligned with the attribute's change through the flow of time. If another agent A_2 is introduced at T_j, and $precedes(T_i, T_j)$ and $precedes(T_j, T)$, and some A_2 attribute affects A_1; the

naive approach will not consider the influence of A_2 on A_1 at time T because A_2 was not present at T, which is the value assumed. With this new policy, in the following definition, I force the search to take such influences into account.

Of course that this new approach, as any one which deals with approximation has its limitations when we have an attribute vary sensible to very small changes. The natural solution would be to add more levels of granularity but this would not leave us free from using approximation at the lower levels. Formally we have the following *solve*/3 definition for this case.

8 - $solve(value(Att, Ag, V) \ @ \ T, \theta_s, \theta_f) \Longleftarrow$

$temp\_subst(T, \theta_s, T_1)$ &

$\neg \ var(T_1)$ &

$body\_of\_simu\_clause(value(Att, Ag, V) \ @ \ T)$ &

$clause(value(Att, Ag, Vi) \ @ \ T_i \Longleftarrow \top :: \theta_i)$ &

$temp\_subst(\theta_i, T_i, T_2)$ &

$\neg \ precedes(T_1, T_2)$ &

$future(T_2, P_a, T_1)$ &

$aligned\_search(P_a, value(Att, Ag, V_i) \ @ \ T_i, \theta_i, value(Att, Ag, V) \ @ \ T, \theta_s, \theta_f)$.

the value of the attribute Att of the agent Ag is V throughout T in θ_s with a set of temporal answer substitution θ_f if, T_1 is the binding of T in θ_s, and T_1 is not a variable, and the goal $value(Att, Ag, V) \ @ \ T$ matches with the body of a simulation clause, and there is a clause $value(Att, Ag, Vi) \ @ \ T_i \Longleftarrow \top$ with TSF θ_n, and T_2 is the temporal substitution of T_i in θ_i, and T_1 does not precede T_2, and T_1 is the future of T_2 after P_a, and the aligned search considering period P_a, the initial clause and its TSF yields a solution for $value(Att, Ag, V)$ at time T, with T in θ_s, and θ_f is its TSF.

Note that the meaning of this clause is similar to the previous one, except that for this case it is necessary to know when the search should stop. Also in the 7th clause, the period assumed between T and T_i will always be some multiple of the scale at which the value of the attribute is changed, while in the second it is not and will affect the

way in which the first value of the attribute will be computed. Thus, $free\_search/5$ is simpler than $aligned\_search/6$ because there is no fixed point to which the search should stop.

For the aligned search the scale of time c_i of the process which changes the attribute is necessary, and also the equivalence between P_a and a time step at level i, i.e. $p(1, c_i)$, and which of them is smaller then the other. In what follows we show the possible cases and their specification, where $change(Att, Proc)$ is true if Att is changed by process $Proc$, $scale(Agent, Proc, C)$ is true if $Agent$ has a process $Proc$ which runs at scale of time C.

1. If $P_a = 0$, then the valid formula has been found and it is just necessary to find a temporal combination.

   $aligned\_search(0, X \ @ \ T_n, \theta_n, X \ @ \ T_s, \theta_s, \theta_f) \Longleftarrow$
   $sub\_tsf((T_s, V_s), \theta_s, \theta_{s'})$ &
   $sub\_tsf((T_n, V_n), \theta_n, \theta_{n'})$ &
   $temp\_comb((T_s, V_s), \theta_{s'}, (T_n, V_n), \theta_{n'}, \theta_f)$.

2. If $P_a > 0$ and it is not equivalent to $p(1, c_i)$, and it is a smaller period than the latter, then a valid formula was found and the current TSF contains correct temporal substitutions.

   $aligned\_search(P_a, X \ @ \ \_, \_, X \ @ \ \_, \theta_f, \theta_f) \Longleftarrow$
   $P_a > 0$ &
   $X = value(Att, Agent, \_)$ &
   $change(Att, Proc)$ &
   $scale(Agent, Proc, C)$ &
   $\neg \ equivalents(P_a, p(1, C))$ &
   $smaller\_period(P_a, p(1, C))$.

3. If $P_a > 0$ and it is equivalent to $p(1, c_i)$, then a forward search from the current time is carried out because the time requested will eventually be reached.

   $aligned\_search(P_a, X_i \ @ \ T_i, \theta_i, X \ @ \ T, \theta_s, \theta_f) \Longleftarrow$

$P_a > 0$ &

$X = value(Att, Agent, \_)$ &

$change(Att, Proc)$ &

$scale(Agent, Proc, C)$ &

$equivalents(P_a, p(1, C))$ &

$search(X_i \ @ \ T_i, \theta_i, X \ @ \ T, \theta_s, \theta_f)$.

4. If $P_a > 0$ and it is not equivalent to $p(1, c_i)$, and this is smaller than P_a, then we have to modularly decompose P_a into a period at the level of c_i. We have two cases as follows.

   1st - If the modular decomposition yields a single period, then a forward search from the current time is called.

   $aligned\_search(P_a, X_i \ @ \ T_i, \theta_i, X \ @ \ T, \theta_s, \theta_f) \Longleftarrow$

   $P_a > 0$ &

   $X = value(Att, Agent, \_)$ &

   $change(Att, Proc)$ &

   $scale(Agent, Proc, C)$ &

   $\neg \ equivalents(P_a, p(1, C))$ &

   $smaller\_period(p(1, C), P_a)$ &

   $mod\_decomposed(P_a, C, P_1)$ &

   $P_1 = p(\_, C)$ &

   $search(X_i \ @ \ T_i, \theta_i, X \ @ \ T, \theta_s, \theta_f)$.

   2nd - The modular decomposition yields a composite period of the form $p(D_1, C_1)$ *plus* $p(D, C)$. In this case, the difference that makes them out of phase (i.e. $p(D_1, C_1)$) will be added to the initial time T_i of the attribute and its value for that time will be assumed to be equal to the initial value. This is the assumption which has to be made that we mentioned before. From this point we start the forward search.

   $aligned\_search(P_a, X_i \ @ \ T_i, \theta_i, X \ @ \ T, \theta_s, \theta_f) \Longleftarrow$

   $P_a > 0$ &

   $X = value(Att, Agent, \_)$ &

$change(Att, Proc)$ &

$scale(Agent, Proc, C)$ &

$\neg\ equivalents(P_a, p(1, C))$ &

$smaller\_period(p(1, C), P_a)$ &

$mod\_decomposed(P_a, C, P_1)$ &

$P_1 = p(D_1, C_1)\ plus\ p(\_, C)$ &

$temp\_comb(T_j, [(T_j, V_j)], T, [(T, p(D_1, C_1)\ after\ T_i)|\theta_i], \theta_j)$ &

$search(X_i\ @\ T_j, \theta_j, X\ @\ T_s, \theta_s, \theta_f))).$

The $search/5$ stops when the the final time is reached, otherwise a new instance of simulation clause is matched with the value of the attribute at the previous time, the constraints are solved and the search re-starts with this new value. Formally we have.

$search(X\ @\ T_n, \theta_n, X\ @\ T_s, \theta_s, \theta) \Longleftarrow$

$sub\_tsf((T_n, V_n), \theta_n, \theta_{n'})$ &

$sub\_tsf((T_s, V_s), \theta_s, \theta_{s'})$ &

$temp\_comb((T_s, V_s), \theta_{s'}, (T_n, V_n), \theta_{n'}, \theta_f).$

$search(X_i\ @\ T_i, \theta_i, X_r\ @\ T_r, \theta_r, \theta_f) \Longleftarrow$

$temp\_subst(\theta_i, T_i, V_i)$ &

$temp\_subst(\theta_r, T_r, V_r)$ &

$precedes(V_i, V_r)$ &

$inst\_simu\_cls(X_i\ @\ T_i, \theta_i, X_f\ @\ T, Constraints, \theta)$ &

$sub\_tsf((T, V), \theta, \theta')$ &

$temp\_comb((T_j, V_j), [\ ], (T, V), \theta', \theta_j)$ &

$((precedes(V_r, V_j)\ \&\ X_r = X_i\ \&\ \theta_f = \theta_i)$

$\vee$

$(\neg\ precedes(V_r, V_j)\ \&\ solve(Constraints, \theta_j, \theta_{j'})\ \&$

$search(X_j\ @\ T_j, \theta_{j'}, X_r\ @\ T_r, \theta_s, \theta_f)).$

If T is not *t-bound*, then the solver simply calls search because there will always be a T'_j such that the $Constraints$ hold. Thus the clause for this case is similar to the first one, except that there is no need to analyse the relation between P_a and the granularity of the process, and $search/3$ is called immediately.

4.7.3 Representing Interaction Between Agents

In what follows, I consider that the attributes of an agent which depend on processes specified at a certain scale of time will have their state changed only at that level. If an agent A_i acting at a coarse level of time interacts with another A_j working at lower level (or at the same level) and from this interaction an attribute Att_j of A_j affects an attribute Att_i of A_i, then A_i has to find out the list of values for Att_j. I will specify how an agent may observe the value of the attributes of other agents if such information is relevant to its behaviour. The *observer process* (OP) will get the progression of the values of Att_j, in a list L, during a specific period of time of the length of k. This idea is depicted in Figure 4.1.

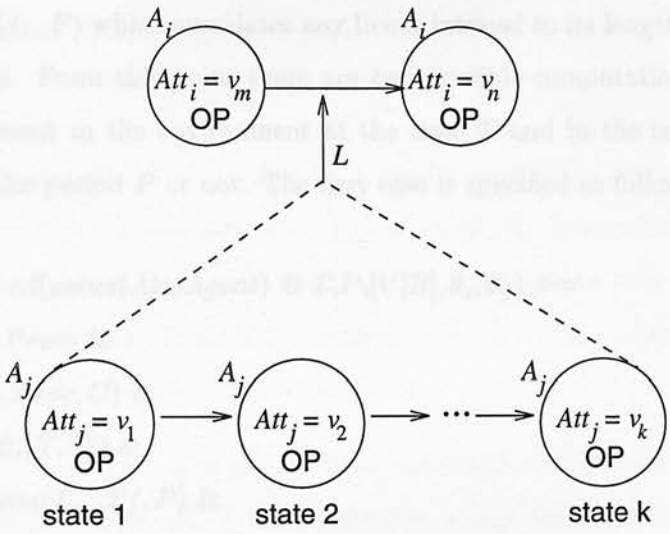

Figure 4.1: Interaction between agents A_i and A_j, where the OP of A_i get the list L of values of Att_j during a period k units of time.

Before we present a detailed specification of the observer process we have to extend the meta-interpreter to deal with the predicate $progress(value(Att, Agent)\ @\ T, P, L)$ which means *the progress of the value of the attribute Att of Agent from T during period P is L*. Why does this need to be specially treated by the meta-interpreter? Because the TSF needs to be carried out over its specification, but a modeller of ecosystems using this language does not need to know how the temporal information is actually handled. We introduce a new clause to *solve*/3 which identifies this predicate and calls the predicate used to define the progress of an attribute during period P. This specification is as follows.

$solve(progress(value(Att, Agent)\ @\ T, P, L), \theta_s, \theta_f) \Longleftarrow$

$progress\_observed(value(Att, Agent)\ @\ T, P, L, \theta_s, \theta_f).$

The specification of *progress_observed*/5, needs information about the process which changes a given attribute, and the corresponding time scale of the agent A_j in order to know how to relate both scales of time. For this, the time interval $T...P\ after\ T$ of observation is open on the right. This means that the value of the observed attribute for the right ending point is unimportant to compute the state of the agent for that time. The reason is because it doesn't make sense to assume the observed value at time $P\ after\ T$ can affect the change of the agent's state for the same time. In order to know the exact point of time in which the observation should stop we use a predicate $length\_of\_time(I_L, P)$ which associates any linear interval to its length (i.e. its ending points inclusive). From this point there are two possible computations. In the first, the agent is present in the environment at the time T and in the second it may be present within the period P or not. The first case is specified as follows.

$progress\_observed(value(Att, Agent)\ @\ T,P,[V|R], \theta_s, \theta_f) \Longleftarrow$

$change(Att, Proc)\ \&$

$scale(Agent, Proc, C)\ \&$

$temp\_subst(\theta_s, T, T_p)\ \&$

$length\_of\_time(T_p...Tf, P)\ \&$

$solve(value(Att, Agent, V)\ @\ T, \theta_s, \theta_n)\ \&$

$compute\_rest(value(Att, Agent, V)\ @\ T, T_f, P, C, \theta_n, \theta_f, R).$

The specification above actually captures the case in which the agent's attribute being "observed" is aligned to the given initial time. However, it may be the case that the initial state of the attribute is either included within the given period (in which case there may be no value for the first time of the period), or the initial state occurs at some time after the last time of observation. To decide in which case the computation should go we just need to find the time of the initial state of the attribute and check it against the limit of observation. This is expressed in the following specification.

$progress\_observed(value(Att, Agent)\ @\ T,P,Prog, \theta_s, \theta_f) \Longleftarrow$

$change(Att, Proc)$ &

$scale(Agent, Proc, C)$ &

$temp\_subst(\theta_s, T, B_p)$ &

$length\_of\_time(B_p...B_f, P)$ &

$\neg solve(value(Att, Agent, \_) @ T, \theta_s, \_)$ &

$solve(value(Att, Agent, V) @ T_i, \theta_s, \theta_n)$ &

$temp\_subst(\theta_n, T_i, B_i)$ &

$(\neg precedes(B_f, B_i)$ &

$Prog = [V|R]$ &

$compute\_rest(value(Att, Agent, V) @ T_i, B_f, P, C, \theta_n, \theta_f, R)$

$\vee$

$precedes(B_f, B_i)$ &

$Prog = [\,])$

In the rest of the progress, represented by *compute_rest*/5, we need the information about the relationship between the period P over which the observation takes place and the scale C of the process that updates the attribute. They can have the same length or one can be greater than the other (two cases). We will write $smaller\_period(P_1, P_2)$ is true if period P_1 is a length of time smaller than P_2.

For the simplest case where P is smaller than the length of C, we have four different cases as depicted in Figure 4.2. First, P starts at a time aligned to an updating time step of the agent's attribute (a). Second, P finishes at an updating time (b). Third, it is in between two consecutive updating time steps, (c), and finally it includes a single updating time step (d).

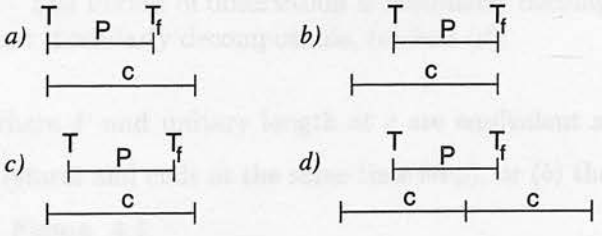

Figure 4.2: Case 1 - a) P starts aligned; b) ends aligned; c) P is included in between two consecutive time steps; d) P includes a single time step.

For any of these cases, the computation of the value of the attribute for the time T_f is

enough. This gives us the following specification.

$$compute\_rest(value(Att, Agent, \_) \ @ \ \_, T_f, p(D, C_1), C_2, \theta_s, \theta_f, [V_f]) \Longleftarrow$$
$$\neg\ (C_1 = C_2)\ \&$$
$$smaller\_period(p(D, C_1), p(1, C_2))\ \&$$
$$solve(value(Att, Agent, V_f)\ @\ T, [(T, T_f)|\theta_s], \theta_f).$$

The meaning for this is *the rest of the values of attribute Att of Agent until time* T_f *considering period of time* D *at scale* C_1, *scale* C_2 *and TSFs* θ_s *and* θ_f *is* $[V_f]$, *if* C_1 *is not equal to* C_2 *and period of time* D *at scale* C_1 *is smaller than period of time 1 at scale* C_2 *and the value of Att of Agent is* V_f *throughout* T *in* $\{(T, T_f)\} \cup \theta_s$ *with a set of temporal answer substitution* θ_f.

The other case is the one in which the length of c is smaller than or equal to the length of P. This can lead us to other possibilities which we classify in two ways. In the first case, depicted in Figure 4.3, P can be modularly decomposed into C and either it matches the actually updating time steps of the agent's attribute (a), or it may be non-aligned (b). It may also not be decomposable and then it matches at the beginning (c) but not at the end, or vice-versa (d).

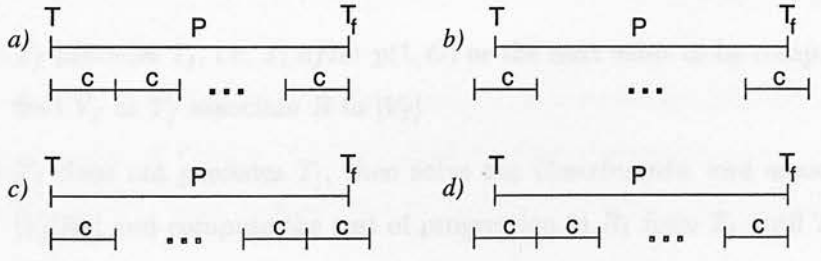

Figure 4.3: Case 2 - The Period of observation is modularly decomposable into C, (a) and (b), and P is not modularly decomposable, (c) and (d).

The final case is where P and unitary length at c are equivalent and either (a) they are totally aligned (starts and ends at the same time step), or (b) they are non-aligned. This is depicted in Figure 4.4.

What has to be done, in the context of *compute_rest*/5, is simply to check if P has a length greater than or equal to the length of c, then it should compute the progression of the value of the attribute until time T_f, taking theses cases above into account. This

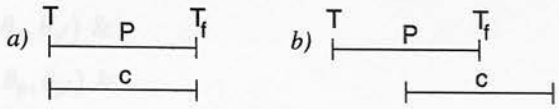

Figure 4.4: Case 3 - *a*) P is aligned to c; *b*) P and c are non-aligned.

is formally specified as follows.

$compute\_rest(value(Att, Agent, V)\ @\ T_i, T_f, p(D, C_1), C, \theta_s, \theta_f, R) \Longleftarrow$

$smaller\_period(p(1, C), p(D, C1))\ \&$

$progression(value(Att, Agent, V)\ @\ T_i, T_f, C, \theta_s, \theta_f, R).$

Now, there are three possibilities when *progression*/4 is called:

1. T_i is equal to T_f, then R should be associated with an empty list.

2. T_i is not equal to T_f, then apply resolution between $value(Att, Agent, V)\ @\ T_i$ and the body of a simulation clause

   $value(Att, Agent, V_j)\ @\ p(1, C)\ after\ T_i \Longleftarrow value(Att, Agent, V_j)\ @\ p(1, C)\ after\ T_i\ \&\ Constraints.$

   Either:

   (a) T_f precedes T_j, i.e. $T_i\ after\ p(1, C)$ or the next value to be computed, then find V_f at T_f associate R to $[V_f]$

   (b) T_f does not precedes T_j, then solve the *Constraints*, and associate R to $[V_j|R_1]$ and compute the rest of progression in R_1 from T_j until T_f.

Considering this, we have the following specification.

$progression(\_\ @\ T_1, T_f, \_, \theta_s, \theta_f, [\,]) \Longleftarrow$

$temp\_subst(U_s, T_1, V_1)\ \&$

$temp\_comb((T_1, V_1), \theta_s, (T2, Tf), [], \theta_f).$

$progression(value(Att, Agent, V_i)\ @\ T_i, T_f, C, \theta_s, \theta_f, R) \Longleftarrow$

$clause(value(Att, Agent, V_j)\ @\ T_j$

$\Longleftarrow value(Att, Agent, V_i)\ @\ T_p\ \&\ Constraints :: \theta_p)\ \&$

$sub\_tsf((T_s, V_s), \theta_s, \theta_{s'})$ &

$sub\_tsf((T_p, V_p), \theta_p, \theta_{p'})$ &

$temp\_comb((T_s, V_s), \theta_{s'}, (T_p, V_p), \theta_{p'}, \theta_{s''})$ &

$temp\_subst(\theta_{s''}, T_s, BT_s)$ &

$next(BT_s, C, T_i)$ &

$(precedes(T_f, T_i)$ &

$sub\_tsf((T_j, V_j), \theta_{s''}, \theta_e)$ &

$temp\_comb((T_2, T_i), [\,], (T_j, V_j), \theta_e, \theta_n)$ &

$solve(value(Att, Agent, V_f)$ @ $T_2, \theta_n, \theta_f)$ &

$R = [V_f]$.

$\vee$

$\neg precedes(T_f, T_i)$ &

$sub\_tsf((T_j, V_j), \theta_{s''}, \theta_e)$ &

$temp\_comb((T_2, T_i), [\,], (T_j, V_j), \theta_e, \theta_n)$ &

$solve(Constraints, \theta_n, \theta_{n'})$ &

$R = [V_j | R_1]$ &

$progression(value(Att, Agent, V_j)$ @ $T_2, T_f, C, \theta_{n'}, \theta_f, R_1))$.

4.8 Application in Simulation Models of Ecosystems

In this section we show how the theory presented in previous sections can be used to model the example of Section 2.3. We shall see how the system works to obtain a deduction for a query. I then discuss improvements of this approach and its limitations.

4.8.1 Representing the Tree and Bug Interaction

The values for growth rate and influence are not realistic but taken just to give us an idea of how an agent's behaviour specification should look like using our formalism. I declare each agent along with the class it belongs to. Each agent has a initial value and we introduce the agent t_2 a bit later in relation to the others. We assume that the growth of all trees are observed on a weekly basis and the bug's movement is daily. The specification for each tree is taken to be as follows, where temporal clauses are in their TNF.

Example 7

$scale(X, growth, week) \Longleftarrow agent(X, tree).$

$change(height, growth).$

$agent(t_1, tree).$

$growth\_rate(t_1, 0.5).$

$value(height, t_1, 9)\ @\ T \Longleftarrow \top :: [(T, t(1,1,1))].$

$value(height, t_1, H)\ @\ T_f$

$\Longleftarrow$

$value(height, t_1, H_i)\ @\ T_p\ \&$

$progress(value(pos, b_1)\ @\ T_p, p(1, week), L_1)\ \&$

$progress(value(height, t_2)\ @\ T_p, p(1, week), L_2)\ \&$

$growth\_rate(t_1, GR)\ \&$

$influence(b_1, GR, L_1, GR_1)\ \&$

$influence(t_2, GR_1, L_2, RealGR)\ \&$

$(H\ is\ H_i + RealGR) :: [(T_f, p(1, week)\ after\ T_p), (T_p, V_p)].$

$agent(t_2, tree).$

$growth\_rate(t_2, 0.6).$

$value(height, t_2, 7)\ @\ T \Longleftarrow \top :: [(T, t(10,1,1))].$

$value(height, t_2, H)\ @\ T_f$

$\Longleftarrow$

$value(height, t_2, H_i)\ @\ T_p\ \&$

$progress(value(pos, b_1)\ @\ T_p, p(1, week), L)\ \&$

$progress(value(height, t_1)\ @\ T_p, p(1, week), L_2)\ \&$

$growth\_rate(t_2, GR)\ \&$

$influence(b_1, GR, L_1, GR_1)\ \&$

$influence(t_1, GR_1, L_2, RealGR)\ \&$

$(H\ is\ H_i + RealGR) :: [(T_f, p(1, week)\ after\ T_p), (T_p, V_p)].$

The $influence/4$ implements how the given agent will affect the tree's growth rate for the given list of values. Say, the specification for the *bug*'s movement is as follows, where $new\_pos/3$ simply changes the bug's position according to its behaviour as specified in the Example.

$agent(b_1, bug).$ $change(pos, movement).$ $scale(b_1, movement, day).$

$value(pos, b_1, 6)\ @\ T :: [(T, t(1,1,1))].$

$value(pos, b_1, PB)\ @\ T_f$

$\Longleftarrow$

$value(pos, b_1, PB_i)\ @\ T_p\ \&$

$value(height, t_1, H)\ @\ T_p\ \&$

$progress(value(height, t_2)\ @\ T_p, p(1, day), L)\ \&$

$average(L, H_2)\ \&$

$AH\ is\ (H_1 + H_2)/2)\ \&$

$new\_pos(PB_i, H, PB) :: [(T_f, p(1, day)\ after\ T_p), (T_p, V_p)].$

A query is transformed into its corresponding TNF before *solve*/3 of the meta-interpreter is called. In Figure 4.5 we show the first steps of deduction for the query $value(height, t1, H)\ @\ t(13,1,1)$. The expressions separated by lines represent the resolution step, except the ones which also have temporal expressions inside boxes. When such expressions are solved the non-temporal part of the expression is solved by standard unification, while the temporal part inside the up and down boxes are unified by temporal combination. The temporal formulae inside boxes are solved subgoals. Their proof trees are similar to this one, and we do not show them for lack of space.

The first line shows the goal in its TNF. In the second line, the initial state of the agent's attribute is obtained (factual knowledge). Next, after detecting that the time requested is specific and that it is non-aligned (the modular decomposition of P_a yields a composite period) wrt the scale of the attribute's update, the length of time for which it is out of phase is added to the initial time. The value for this time is then assumed to be the last one.

The third step (shown in the partial proof tree) uses the assumption made and matches it against the body of the second clause for the agent's attribute (in this case *height* of t_1). This is represented on the top-left subgoal. The second subgoal uses only standard unification, although it has temporal expressions as one of its terms they are to be used for guiding the temporal deductions of the observer process. The result of this is inside the dashed box down on the left, where the expressions inside are the solved temporal subgoals for each time step inside the period of one week. The

Goal - $value(height, t_1, H)\ @\ T :: \theta = \{(T, t(13,1,1))\}$

A1 - $value(height, t_1, 9)\ @\ T_1 \Longleftarrow \top :: \theta_1 = \{T, t(1,1,1))\}$ (Factual Knowledge)

A2 - $value(height, t_1, 9)\ @\ T_2 :: \theta_2 = \{(T_2, t(6,1,1))\}$ (Aligned search assumption)

A3 - 2nd clause + A2 of $value/3$ as follows $\theta_3 = \{(T_f, p(1, week)\ after\ T_p, (T_p, X)\}$

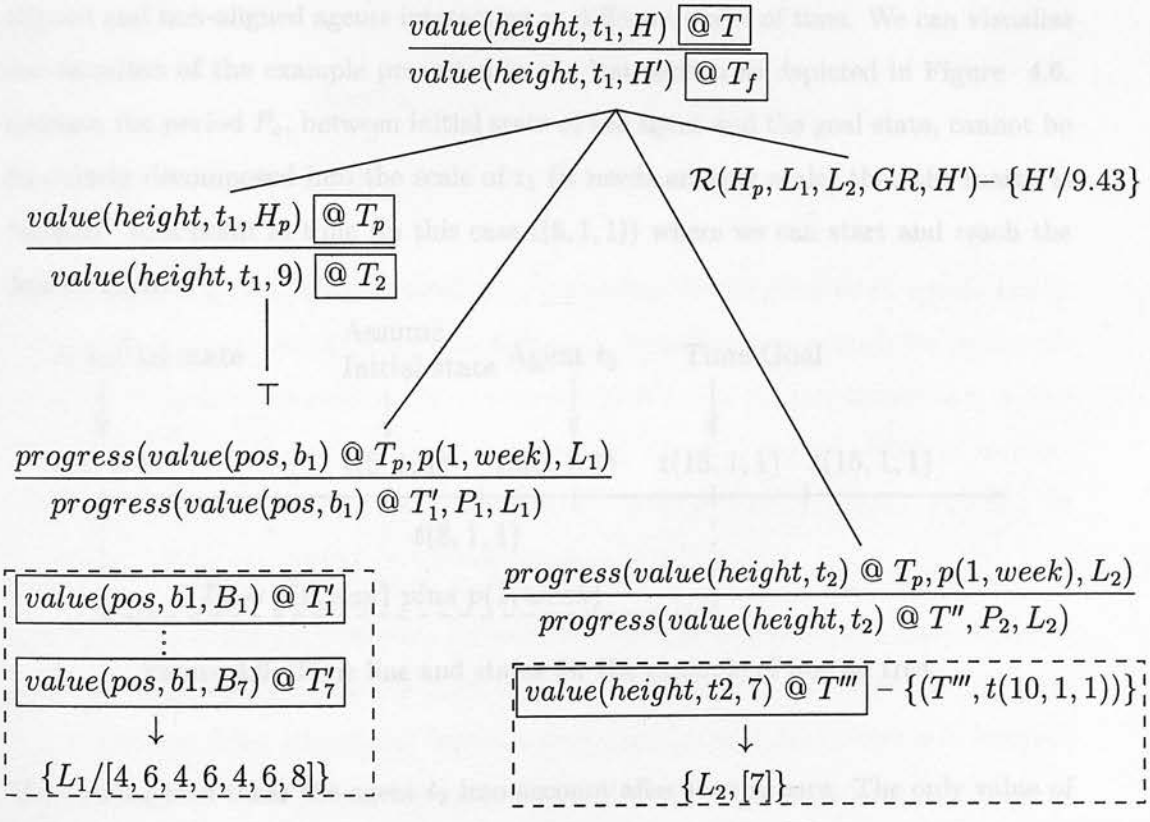

Figure 4.5: Partial proof tree for the goal $value(height, t_1, H)\ @\ t(13,1,1)$.

arrow points to the substitution found for L_1. The next subgoal is another call for the observer process, which returns the progress of t_2 as a substitution. Note that the value found corresponds to exactly the time of t_2's entrance into the environment. This corresponds to Case 3.b explained in Figure 4.4. The last goal actually represents the sequence of goals which compute the new value of the attribute requested by the top goal.

4.8.2 Discussion

The approach we have just seen provides a mechanism for problems where there are aligned and non-aligned agents interacting at different scales of time. We can visualise the situation of the example presented in the last section as depicted in Figure 4.6. Because the period P_a, between initial state of the agent and the goal state, cannot be modularly decomposed into the scale of t_1 (it needs another scale) then the search is "shifted" to a point in time (in this case $t(6,1,1)$) where we can start and reach the desired state.

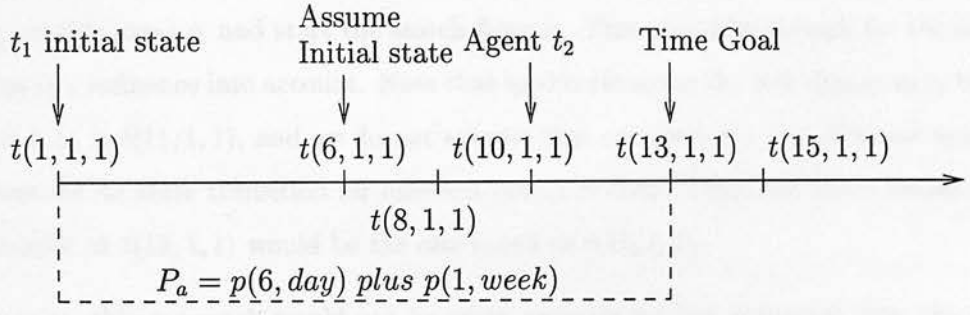

Figure 4.6: Time line and states for the example of bug ad tree.

This assumption takes the agent t_2 into account after its entrance. The only value of t_2 that will affect both t_1 and b_1 between $t(6,1,1)$ and $t(13,1,1)$ will be initial state, with later values of $t-1$ having no influence. Suppose the goal state was a later time, say $t(20,1,1)$, then for the period between $t(10,1,1)$ and $t(17,1,1)$ the influence of t_1 upon t_2 would be from two values, for t_1 between $t(13,1,1)$ and $t(20,1,1)$ would be two values as well. However, how would the accuracy of this approach be affected in a case where we had other agents initially present at different time stamps?

Now, suppose we had two other agents, say t_3 and t_4, each introduced into the envir-

onment at times $t(4,1,1)$ and $t(12,1,1)$, respectively. The goal is still the same, i.e. $value(height, t1, H)$ @ $t(13,1,1)$, and both affect t_1 and t_2. What should be considered the initial state for the search? This situation is depicted in Figure 4.7. According to the adopted policy the initial state considered would still be $t(6,1,1)$ and the value of t_1's height assumed for this time would the one for it actual initial state. This is not fully accurate because t_3 will have affected t_1 since $t(4,1,1)$.

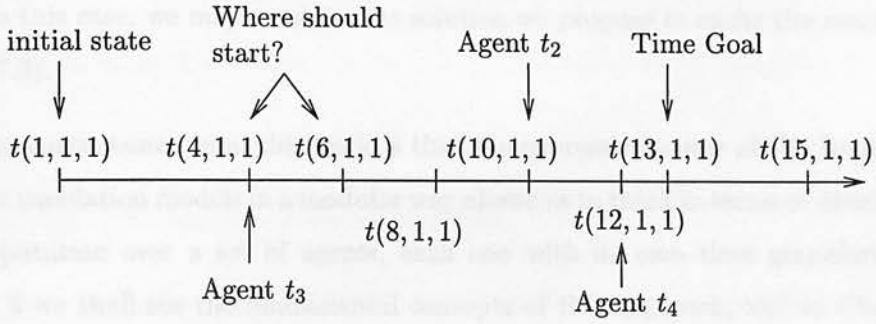

Figure 4.7: Another situation in which two other non-aligned agents are present.

An alternative approach could be to take the earliest initial state of all agents affecting (in this case) t_1 and start the search from it. This would be enough for the search takes t_3's influence into account. Note that in this situation the last change in t_1 before $t(13,1,1)$ is $t(11,1,1)$, and we do not assume that an agent has two different specifications for its state transition for different grains of time! Thus, the value assumed for t_1 height at $t(13,1,1)$ would be the one found at $t(11,1,1)$.

However, this approach would not be more accurate (or less accurate) than the one I proposed. The reason is that by doing this we would not take the influence of t_4 upon t_1 into account. This other agent has been introduced at $t(12,1,1)$ which is in between $t(11,1,1)$ and $t(13,1,1)$. Looking more deeply this approach is symmetric to mine in the sense that I apply a policy of last state of agent in the beginning and this one in the end.

The other aspect is the interaction between aligned and non-aligned agents at different scales of time. The new abstractions and the new policy for non-aligned interactions provide a mechanism for problems of this nature. The accuracy of the approach, however, is compromised when a number of non-aligned agents are introduced into the environment.

The use of forward chaining avoids the re-computation of pure temporal expressions already computed by the recursive definition, but only works for a restricted form of clause. However, even though this approach is better than unconstrained search backwards through the flow of time, some inefficiency is inevitable because each agent's behaviour only manages its own values. No restriction is imposed on when during the interaction others may ask for previous values, in which case they have to be computed again. In this case, we may combine the solution we propose to cache the results (see Section 7.3).

The main consequence from this work is that the expressive power of the language to represent simulation models in a modular way allows us to think in terms of distributing the computation over a set of agents, each one with its own time granularity. In Chapter 5 we shall see the fundamental concepts of this approach, and in Chapter 6 we will see an architecture which implement them.

4.9 Limitations of NatureTime for Large-Scale Simulation

One of the main limitations of using a traditional AI based approach for modelling ecosystems is the inefficiency of its use for large scale knowledge bases. The complexity of query increases exponentially as a consequence of the increase in the chain of dependency among rules. In the case of simulation models this becomes unreliable even for simple applications such as the models for **Example 1**. Because actual ecosystems are inherently parallel, it would be more natural if the models we build to represent them could also present the same feature. Note that such a limitation also raises in the another approach mentioned in the last section for taking the earliest initial state of all agents.

A natural suggestion is to distribute the computation in order to gain a significant reduction in complexity by imposing some useful structure on the problem solving process. Though one might propose that a parallel version of **NatureTime** could solve the problem, it is very unlikely that this would help handle more complex ecosystems. There is no way to guarantee that in every new extension of the ecosystem the model

would run soundly.

We can achieve such a level of modularity only by means of a scheme such as the GSCS (**Definition 2.1**). However, such a scheme does not say anything at all about how to represent the influences of an agent, how they change through time or which mechanisms are necessary to an agent get the information it needs from others in order to compute their influence upon its own behaviour and attributes.

The conclusion from this is that the computational entity being modelled should be able to

1. run with or without the presence of others,
2. interact with other agents in order to obtain information they need (traditional AI KBS enhanced with communication capabilities),
3. coordinate its actions with others according to the level of granularity of each of its state transition process,
4. react properly according to the influence other agents may exert upon its behaviour.

Along with these features, it is desirable that such an architecture is endowed with proper structures to take advantage of the hierarchical partition that some problem domains have, and so obtain an understandable chain of dependency among agents and group of agents.

4.10 Summary

- The Temporal Substitutional Framework proposed here separates standard and temporal unification. This allows a clear and consistent treatment for shared temporal variables.
- The mechanisms introduced for dealing with aligned and non-aligned interacting agents are very powerful.

- The policy adopted to get the last state of an agent for time stamps in between two consecutive time steps is symmetric to the one which updates the initial state of all agents every time a new agent is considered. Analysis suggests that the limitations of both approaches arise out of the fact that they simulate groups of agents in a single computational model.

- The expressive power of the language to represent simulation models in a modular way allows us to think in terms of distributing the computation over a set of agents, each one with its own time granularity.

Chapter 5

A View of Agents in Ecological Modelling

5.1 Introduction

In Chapter 4 we saw a framework to deal with temporal substitutions for **NatureTime** logic, and I showed how to implement the logic and use it to represent simulation models. The modularity of the specification suggests that the computation of an agent's behaviour (or the execution of its actions) can be distributed. For this, new mechanisms for representing the interaction between agents must be introduced, so that they may have a degree of autonomy and react properly according to the influence other agents may exert upon their behaviour.

In this chapter we shall see what I call an *ecoagent*, which is a building block for the development of simulation models of ecosystems in a multi-agent architecture. Throughout the rest of this thesis I will use this term with this meaning, and I will use the term "agent" in this sense. Also, by "environment" of an agent I mean a virtual environment (possibly composed of inter-connected computers), where the agent senses actual influences over its behaviour.

I argued in Section 1.1 that traditional simulation models do not consider human actions as part of the model. Here I assume that humans are agents of the environment, and so their actions should be modelled in some way. It will be clear during my exposition where the concepts of cognitive or rational agents (for humans) is allowed to interact with other kinds of agents.

5.2 Ecological Agent

The word agent, as considered in this work, is the thing that exerts power or produces an effect or change, or acts for another [1]. From this point of view, an ecological agent, simply *ecoagent*, may be associated with natural resources such as water or nutrients, to plants and trees, to animals, and humans. An *ecoagent* is defined by a set of attributes and processes to represent its behaviour. Such attributes can be static or dynamic, and internal or external.

Static attributes are time independent properties while dynamic attributes are time dependent. Internal attributes are not "visible" to the external world but other agents may know or ask about them. Whether or not the required information is supplied depends on the kind of agent trying to obtain such an information. On the other hand, external attributes are externally visible to others.

The issue now is how can we classify ecoagents according to their functionality within the environment. The purpose of such a classification is to identify the kinds of interaction between agents of certain classes, and among agents of different classes. By doing this we will be able also to determine the range of actions they can perform, and thus to classify the sorts of messages agents exchange to perform such interactions. Furthermore, we will be able to envisage dependency relations between classes of agents and actions within an agent or group of agents.

Agents are grouped into classes according to their behaviour. The behaviour of an *ecoagent* is manifested by the actions it takes upon the environment and also upon its attributes by means of its internal processes. An action of an agent may succeed or fail and affects the environment through messages it sends to the subject of its action. As these messages are similar to the ones used in theories of multi-agent systems based on the *speech act* approach [Austin 62], for example [Cohen & Levesque 90, Shoham 93, Singh 94, Barbuceanu & Fox 95], then we may also assume that actions in this work are represented as speech acts.

Changes to an agent's attributes are achieved by the agent sending messages to its in-

[1] The Concise Oxford Dictionary - 7th Edition, 1995

ternal processes (or actions). The behaviour of an agent is affected by the environment depending on the contents of the messages it receives from it. The kinds of message an agent is allowed to send and receive from the environment characterise it as follows.

- A *Passive* agent does not exert actions upon others but supplies them with something. Thus such an agent never sends requests or queries to other agents. It only receives messages and is only able to send answers to those requests or queries it receives. Agents of this class represent things which offer pools of resources to other agents. For instance, soil can be modelled as a passive ecoagent with pools of water and nutrients to offer.

- *Reactive* - the agent may receive and send requests, queries, and information about its internal state and situation in relation to the group to which it belongs. Such an agent usually depends on the presence of *passive* agents as resources for its internal processes, as well as the presence of others of its class. Examples of things which might be modelled as reactive are: a tree, a swarm of bugs, a wolf or a rabbit.

  If necessary this class may be sub-divided in agents which have the ability to move around or need a mate to reproduce, and those agents which lack such abilities or necessities. For example, a plant and an animal may be classified in this way. However, from an ecological point of view a plant (or tree), depending on the species, may be modelled as an agent which needs another individual to reproduce or present some form of movement. This debate is not in the scope of this work and I leave this decision for the modeller.

- *Active* - the agent may, along with the other kinds of messages, order some action upon other agents. Among individuals of this class an agent may or may not attend queries and requests, and may send requests, queries and information which are not necessarily related to the messages it receives but to its internal knowledge such as beliefs, desires, etc.But even in this case, the agent sends message to express the effects of its action.

Thus, the classification of an *ecoagent* is determined by its reasoning capabilities, its level of self awareness and how constrained are its abilities to transform the environ-

ment by means of its actions. The philosophy behind this concept is that a passive ecoagent when enhanced with autonomy, the ability to send queries, requests and certain kinds of commands becomes a reactive ecoagent. Analogously, a reactive *ecoagent* becomes an active one when extended with self awareness and more ability to change the environment. This is depicted in Figure 5.1, where the vertical arrow says that the wider the scope of actions of an *ecoagent* the more self aware is the agent.

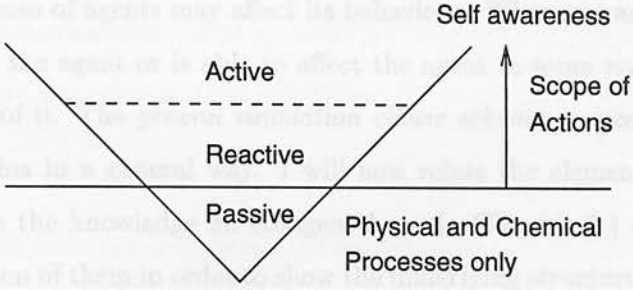

Figure 5.1: Hierarchy of action and self awareness among agents.

Of course, within living creatures, there are those with limited forms of actions and those with rich forms of expression. Hence, the divisions are much more difficult to draw biologically than they are in an agent architecture. Independent of the kind of ecoagents a modeller is going to use for representing his/her view of an actual ecosystem, each agent should present the properties pointed out in Section 4.9.

It would be ideal if we could leave the modeller only with the task of specifying the knowledge associated to the behaviour of the *ecoagents* she/he wants to develop. In other words, the modeller does not need to know how the computation is distributed and its underlying structures or how the mechanisms for coordinating concurrent processes are implemented. This is an issue of architecture which we shall address in Chapter 6.

The rest of this chapter is dedicated to the description of the knowledge associated with *ecoagent* specification, actions of agents, group of *ecoagents*. By this I mean the set of attributes which characterises *ecoagents*, individually, and groups of them; what affects them, how and under which circumstances; the actions an agent may perform, and possibly the order and scale of time of such actions. Finally, I shall discuss related work in distributed artificial intelligence.

5.3 Knowledge Associated with Ecoagents

In Section 2.4 I suggested that it is necessary to represent all potential sources of influence upon an agent's behaviour. This does not mean an agent needs to have a complete representation of the world and that it knows about every other agent. But at least an *ecoagent* should be endowed with knowledge about the circumstances under which certain classes of agents may affect its behaviour. Whenever an instance of such classes is near to the agent or is able to affect the agent in some way, then the agent should be aware of it. The *general simulation clause schema* proposed in Section 2.4 formalises this idea in a general way. I will now relate the elements and structures a GSCS needs to the knowledge an ecoagent has. In Chapter 6 I shall give a more detailed description of them in order to show the underlying structures which represent such local knowledge and the inference mechanisms.

5.3.1 Potential Influences

This is a set of meta-relations about the potential sources of influences among agents from different classes. For every *ecoagent* there will be set of classes of agents that may affect it. I assume that such knowledge is time independent because it does not represent the actual influences over time. Actually, it represents only what sort of influence may affect the behaviour of an agent when instances of those classes are present within the agent's local environment.

There are two basic ways in which an attribute of an agent may affect another agent's process. First, through the value of the attribute. Second, if the agent absorbs a certain amount X of the value the attribute. While in the former case the value of the attribute exerts influence on the process, the latter introduces the notion of a transfer of substance from the resource available. Although the latter can be seen as a special case of the former, this difference will be stressed for representational purposes. Formally, this can be described as follows.

Definition 5.1 (Potential Influence) *Let A be an agent, P_i be the name of a process, Att_j be an attribute of a class C of agents. To say that P_i of A is affected by*

either the value of Att_j of class C or the amount X of Att of class C absorbed by A, under condition $Cond$, and that this affect is computed by function π such that Att is the kth parameter of π, I shall write $affect(P_i, A, Value, Cond, Func)$, where

- *$Value$ can be either*

  $value(Att_j, agent(C))$ *or*

  $absorbed(amount(X, Att_j), agent(C))$

  Let Σ_v be the set of terms with this form.

- *$Cond$ can be either*

  1. – $holds(loc\_env\_relation(LocEnvRel, A, agent(Cls)))$ *or*

     – $holds(loc\_env\_relation(LocEnvRel, agent(Cls), A))$ *or*

  2. – $holds(Relation, value(Rel, value(Att_j, agent(C)), value(Att_i, A)), Const) \rightarrow LocEnvRel$ *or*

     – $holds(Relation, value(Rel, value(Att_i, A), value(Att_j, agent(C))), Const) \rightarrow LocEnvRel,$

  where Att_i is an attribute of A involved in the potential interaction, $LocEnvRel$ is the name of a relation with respect to the agent's local environment. I call this kind of relation a local environmental relation. *(Let Σ_h be the set of terms of this form.)*

- *$Func$ can be either*

  – *constraint - to mean the condition is a necessary constraint to local environment relation to be true. This case is only applied to second kind of $Cond$ above.*

  – *$func(\pi, n, k)$, where π is the name of a function with n parameters to compute Att_j's influence, and, in this case, Att_j is the k_{th} parameter of π*

  Let Σ_{inf} be the set of terms of the form $func(\pi, n, k)$.

Note that $func(\pi, n, k)$ does not say how the interpretation function works because it is domain dependent, and π is just the identification of the function. Thus there

may exists different ways to be specified. This term is a kind of protocol interface for whatever the language used to define it.

When an agent notices the entrance of a new agent into the environment (via a sensing message), or another agent changes its location the agent makes an analysis of the agent's features against its potential influences in order to detect whether or not the new (or moving) individual will affect its behaviour. But it will only use the function to compute the influence if the condition under which the influence takes place holds, and the agent has obtained the required value. The interpretation an agent gives to the condition for the influence is that:

- the simple case of *Cond* is found, the interpretation is that *the value should be obtained via a sensing message.*
- the rule case of *Cond* is found, the value should be obtained from the agent's current local environment, i.e. an internal structure which represents the local environment.

5.3.2 Local Environment

As we saw above, that when defining influences from the environment one should care about only information which may actually exert some power on the agent's behaviour. From such a information an *ecoagent* is able to derive the actual influences during the period of time in which they take place. This is part of the agent's *local environment*. A local environment is a set containing information about the instances of classes of influences (basically agent identification) along with the nature of the relation and some possible value associated with it. This means that an agent is aware only about the current things happening nearby or those which are internal to it (as in the case of active *ecoagents* which are affected by non-physical things such as beliefs).

The purpose of representing an agent's local environment is to reduce the space of possible relationships between an ecoagent and the environment. As an *ecoagent* can be anything in a given environment, then the modeller must specify the sorts of valid relationships among classes of ecoagents. Depending on the kind of ecoagent there may exist different sorts of local environmental relationships (I may refer to these just as

local relations). In what follows I list some possible examples of what I mean by local relationships for each kind of ecoagent. Note that this is not meant to be an exhaustive and precise ecological classification which is impossible to do.

- *Passive* - Depending on the kind of passive ecoagent there can be many possible relations. For example, a soil may have trees as clients for water and nutrients, an active agent as owner, etc.

- *Reactive* - Among the possible classes of this kind of ecoagents this is work is particularly concerned with
  - plants and trees - usually the sorts of local relations are: 1) environmental stress - when other agent interferes on the plant ability to acquire resource (light and nutrient) [Mix *et al.* 92]. The modeller has to give the other classes of ecoagents that may cause such interference and the conditions for this take place. In [Niven 82] these are called *malentities*. 2) environmental resources - are all other agents that may be source of resource: soil water and nutrients. In [Niven 82] these are called as *resources*.
  - Animals - common relations within this class are: prey, predator, mate, (which may raise) competitors, offspring and malentities. It is a task for the modeller to identify the classes of ecoagents which fall into these categories in relation to others.

- *Active* - Depending on the application domain there can be a potentially infinite number of relations and more complex expressions might be necessary to represent *Cond* in the meta-predicate $affect/5$ (Section 5.3.1). Some examples of relation from **Example 1** are:
  - Social relations - relations such as friends, relatives, neighbours, etc.(*Angelina*, *Elza* are examples of friends). They influence an agent's action because of the beliefs (or other mental states) an agent has about them.
  - Job relations - the relation between *Minor* and the organisation he works for, or between *Elsa* and the *Buyern* company. They can be: boss, employer, manager, director, subordinated. Here, the influences upon an agent's action may be related to commitments and goals.

- Legal relations - "power" relations between *active* and the other classes of ecoagents. For example, *John* is the Landlord of the place where the insect pest attacked. *Minor* works for the government and he has a range of autonomy to execute some actions against those who do not follow the rules established for respecting the protection of the environment.

An agent's local environment is a dynamic structure, generated according to the instantiation of potential influences affecting an agent. Two things are necessary for it: a language to represent such knowledge and specialised inference mechanisms. The language should be able to represent schemas of reasoning about local relationships, functions to compute influences and state transitions as proposed in Section 5.3.1. By accessing such schemas the agent is able to detect whether a new agent is or not a potential influence and if it should be part of its local environment or not.

5.3.3 Sensing, Information Resource and Assimilating

Sensing is the ability of an *ecoagent* to capture changes in the environment which may affect its local environment, and also to acquire resource information from the local environment. In the first one the agent interacts with the environment and accesses its potential sources of influence in order to figure out the agents which compose its local environment. The events associated with it are the entrance of the agent itself into a new environment, and the entrance or departure of agents from its local environment. This can be formalised as follows.

Definition 5.2 (Sensing External Environment Ability) *Let Λ be set of possible states of an agent A, Ψ be the set of possible states of external information, and Σ_Θ be the set of possible local environments of A. Then,* sensing external information ability *is defined by a function $\mathcal{S}_\Theta : \Psi \times \Lambda \rightarrow \Sigma_\Theta$.*

Once an ecoagent has knowledge about its local environment it will be able to know what kinds of resources there exist to suit its behaviour. As the environment is dynamic it is expected that an agent's local environment also changes as time goes by. An agent uses its knowledge about potential sources of influence to "sense" such changes in the

environment. This will allow an ecoagent to update its local environment according to the classes of agents which become or are no longer a source of influence.

The kind of knowledge an agent has to acquire I call an *information resource*, which can be a value (numerical or not) or a tuple composed of numerical, qualitative or other tuples as values. The information resource, however, is not necessarily the same as the external information. This will depend on how the agent interprets or filters the external information and transforms it into its internal knowledge. Such interpretation can be related to internal capabilities, physical devices (if it is the case), or beliefs about the world. Thus, sensing is actually a mapping from the information as it is presented by the environment to a useful or meaningful value for the agent. A good example of it is a thermostat which transforms the kinetic energy captured into a number to which we give the interpretation of temperature. This can be formally described as follows, where the state of an agent is represented by the value of its attributes (whatever their type).

Definition 5.3 (Sensing Information Resource) *Let Λ be the set of possible states of an agent A, Ψ be the set of states of external information exerting some influence on A, and Σ_Γ be a set of sets of information resource for A.* Sensing information of a resource *is is defined by a function $\mathcal{S} : \Psi \times \Lambda \rightarrow \Sigma_\Gamma$.*

After transforming external information into an information resource, an agent must assimilate it in some way. This is related to the state of the agent and how it will compose the set of influences upon a given attribute during a certain period of time. Thus, I assume that agents have a sort of assimilation process responsible for transforming information resource into meaningful properties for its actions. As its actions can perform changes over its attributes, I call the result of such assimilation an *attribute change factor*.

Definition 5.4 (Assimilation Resource of an Ecoagent) *Let Λ be the set of possible states of an agent A, Σ_Γ be a set of sets of information resource, and Υ be a set of attribute change factors or information resource. Then* assimilation resources *of an agent is defined by a function $\Re : \Lambda \times \Sigma_\Gamma \rightarrow \Upsilon$.*

Again we have a general definition and the exact structure of Υ is unknown. Depending on the type of the information resource acquired, a change factor can be a numerical value, a tuple of values or even an internal or external action. This is a domain dependent issue and there is no framework which provides a general assimilation function. In Chapter 6, we shall see an architecture which implements the ideas of this chapter and how this problem is tackled.

5.4 Actions of Ecoagents

The actions an agent is able to perform (according to its class) are those to change its state, the state of the environment or to coordinate its actions. In the first two, other agents may observe the actions unless they change internal attributes of the agent. By internal attributes I also assume knowledge about beliefs, intentions, desires, etc. As far as change in the environment is concerned, only those actions which perform changes on external attributes are directly relevant. Changes on the internal state of an agent are useful either when it is interacting with others or reasoning about others' internal state, or reasoning about its own state in order to take some decision.

Usually, changes are considered as functions which map the value of the attributes of an agent from one state to a subsequent one. Such functions should take the influence of the environment into account. Recall that an agent has a set of potential influences (see Section 5.3.1), from which it is able to conclude whether or not environmental information affects its behaviour.

Definition 5.5 (Action of an Ecoagent - I) *Let Λ be the set of possible states of an agent A and Υ the set of information resource of A. An* action *of A is a mapping $\mathcal{A}$ from $\Lambda \times \Upsilon$ to Λ, written $\mathcal{A} : \Lambda \times \Upsilon \rightarrow \Lambda$.*

It is not difficult to see that an action is a composition of sensing, assimilating and executing the action in itself, i.e. $\mathcal{A}(\Lambda, \Re(\Lambda, \mathcal{S}(\Lambda, \Psi))) \rightarrow \Lambda$. Note that this definition generalises even those actions in which no sensing or assimilating seems to be involved. In these cases the element of Ψ to which the function is applied is the empty set.

Figure 5.2 shows a graphical view of this definition, where s_{t_i} and s_{t_j} are the sets

$Att_1, \ldots, Att_n$ and $Att'_1, \ldots, Att'_n$, respectively; and Ψ is $Inf_1, \ldots, Inf_k$; and the set of actions are represented by $\mathcal{A}_1, \ldots, \mathcal{A}_n$, each $\mathcal{A}_i$ for each attribute Att_i. In this picture, I show several actions which may be taken in parallel from one state to another, assuming that there is no dependency relation among them. I will show that they do not need to be specified at the same scale of time.

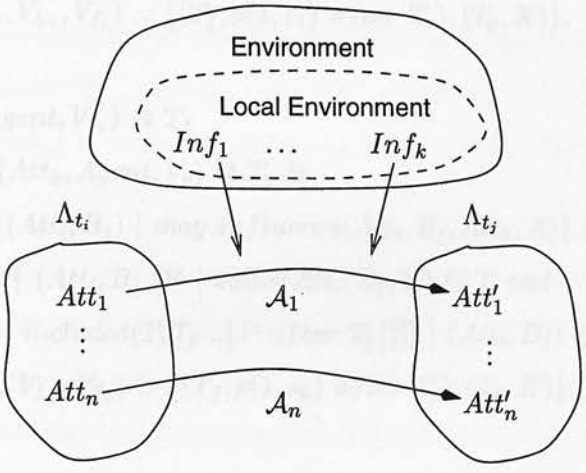

Figure 5.2: State transition of an agent.

The set of actions of an agent, no matter its class or which attribute they may change, could be specified by a set of *general simulation clause schemas* as proposed in Section 2.4. But here, instead of $influence/4$ we use $may\_influence/4$ to better represent a potential influence. I then propose a definition of an *ecoagent* by using the previous concepts along with the specification of its behaviour. In what follows I assume the existence of some framework for agent attribute definition (name, type of attribute, etc.), actions an agent can recognise and send to the environment, and I mark part of the actions specification with Ⓐ and Ⓑ to explain them afterwards.

Definition 5.6 (*Ecoagent* in terms of GSCS) *Let* $ecoagent(\alpha, Cls)$ *be the declaration that* α *is an agent of class* Cls, $Att_1, \ldots, Att_n$ *be the attributes of* α, $P_1, \ldots, P_k$ *be processes responsible for changing* k *dynamic attributes of* α, $\mathcal{S}_\Theta$ *be the agent's ability to sense the environment,* $\mathcal{S}$ *be the agent's ability to sense information resource,* $\Re$ *be the resource assimilation function of* α, *and* $\mathcal{A}_{ext}$ *be the set of actions the agent recognise from the environment. Then, an* α *is an* ecoagent *if along with these information its set of processes* $\mathcal{A}$ *can be defined as follows*

$$
\begin{aligned}
P_1 : & \; value(Att_1, Agent, V_{f_1}) \; @ \; T_f \\
& \Longleftarrow value(Att_1, Agent, V_p) \; @ \; T_p \; \& \\
& \textcircled{A} \quad I_1 = \{(Att_i, B_1) \mid may\_influence(Att_i, B_1, Att_1, A)\} \; \& \\
& \qquad V_{L_1} = \{(Att_i, B_1, [V \mid value(Att_i, B_1, V) \; @ \; T \; and \\
& \qquad\qquad included(T, T_p ... \lfloor P \; after \; T_p \rfloor)]) \mid (Att_i, B_1) \in I_1\} \; \& \\
& \textcircled{B} \quad \mathcal{R}_1(V_p, V_{L_1}, V_{f_1}) :: \{(T_f, p(1, s_1) \; after \; T_p), (T_p, X)\}. \\
\vdots \quad & \quad \vdots \\
P_k : & \; value(Att_k, Agent, V_{f_k}) \; @ \; T_f \\
& \Longleftarrow value(Att_k, Agent, V_p) \; @ \; T_p \; \& \\
& \textcircled{A} \quad I_k = \{(Att_i, B_j) \mid may\_influence(Att_i, B_j, Att_k, A)\} \; \& \\
& \qquad V_{L_k} = \{ \; (Att_i, B_j, [V \mid value(Att_i, B_j, V) \; @ \; T \; and \\
& \qquad\qquad included(T, T_p ... \lfloor P \; after \; T_p \rfloor)]) \mid (Att_i, B_j) \in I_k\} \; \& \\
& \textcircled{B} \quad \mathcal{R}_k(V_p, V_{L_k}, V_{f_k}) :: \{(T_f, p(1, s_k) \; after \; T_p), (T_p, X)\}.
\end{aligned}
$$

In the specification above, Ⓐ is dependent on interaction with the environment. The search for such values of other agents' attributes (a deduction in a single computational model) is obtained by communication via message-passing. But it also depends on how the agent interprets the values of such influences, i.e. how such values interfere on the change of each Att_i or (using human like notions) the beliefs A has about the world (see Section 5.3.3). On the other hand, Ⓑ depends on the way the agent assimilates what it has interpreted from the external environment and how this is used to execute the changes over its attributes. In other words it is related to Know-How, Intentions and Commitments.

Also the influence of the assimilation factor on the process of acquiring resource may determine the actions the agent needs in order to obtain such resources. This may cause interaction between internal processes. To introduce such computational structures in the language will force a modeller to think twice in terms of computation. The first consideration is the causal relationship between the processes of the environment. The second is related to the causal relationship between actions represented by means of messages. As the purpose of this new representation is to distribute the computation, the non-determinism introduced by the order in which messages may arrive raises more

complexity in the system development.

However, instead of providing such structures to a modeller it would be more interesting if we could offer schema for representing all this knowledge. The underlying interconnection between concurrent processes and how the agent deals with messages would be no problem. The modeller would be free to concentrate her/his efforts on modelling the problem, without concerns about more complex computational problems such as deadlocks. The execution of such models should have the same outcome as those specified using a set of GSCS in which one would have to care about these complex matters.

For this it is necessary that each agent knows about which action changes which attribute. After this, for each of these actions, how they should be computed, i.e. which function is to be used to perform such a change. Note that the relation between attributes or internal actions can be specified in such a computational function definition. This and the details of an agent's internal computation are matters for the next chapter. The kind of information which is needed is related to the scale of actions, the ordering relationship among them and how a group of agents should be treated.

5.5 Scale and Ordering Relations of Ecoagent's Actions

The ordering of processes is related to the causal relationships between actions. Inherent to the concept of causality is the concept of time granularity of each process. This will influence the ordering relationship that an agent needs to impose over the set of its actions. The scale of processes is declaratively defined as we saw in Section 4.7.2 when we defined the possible cases for *aligned_search*/6. Here, we may use just the name of the process and of the scale.

An agent needs to order its processes to synchronise the changes they produce over its attributes through the flow of time of the overall system, also known as global time. For this, each agent has intrinsic and internal knowledge of its process ordering relationship (POR). Such knowledge is obtained from the definition of scales of time of each process presented above.

Definition 5.7 (Temporal Granular Scheduler) *Let Λ be the set of possible states of an agent A, n be the number of processes of A, $\mathcal{P}_s$ be a set of pairs (P_i, s_i), $i = 1, \ldots, n$, where each P_i is a process of A and s_i the time scale of P_i, and $\mathcal{T}_M$ be the set of global time of the environment. Then, Π_s is a* Temporal Granular Scheduler *defined as function $\Pi_s : \Lambda \times P_s \times \mathcal{T}_M \rightarrow \mathcal{T}_{GS}$, where each element of $\mathcal{T}_{GS}$ is in the form*

$$\langle (T_1, \{P_1; \ldots; P_i\}), \ldots, (T_n, \{P_k; \ldots; P_n\}) \rangle,$$

where each $(T_i, \{P_j; \ldots; P_{j+m}\})$ means that at time T_i all $P_j, \ldots, P_{j+m}$ can be executed in parallel.

Note that this definition is not committed to any specific time representation, except that the time framework used should allow granularity of time. Also, the fact that a process is allowed to be executed does not mean it is going to succeed or satisfy its intended goal. In this case the failed process may enter again into $\mathcal{T}_{GS}$.

However, this scheduler does not represent the other important aspect which is the causality relationship. In other words a scheduler or plan of action has to consider two things:

1. time scale of change - $\langle \overbrace{(P_{t_1}; \ldots; P_{t_i})}^{S_1}, \ldots, \overbrace{(P_{t_k}; \ldots; P_{t_n})}^{S_n} \rangle$
2. causal relationship among processes - $\langle (P_{c_1}, \ldots, P_{c_i}), \ldots, (P_{c_l}, \ldots, P_{c_m}) \rangle$

These two lists also represents the ordering of the processes. In S_1 they all have the same scale of change s and do not have any causal relationship among them. In the second, they may not have the same scale of change but are independent from one another. Because (2) is a "strong scheduling policy" it may cause processes on the top of (1) not to be executed, and so postponed for some time later. This work is only concerned with the granularity of such actions, but other constraints on planning an agent's actions can also be considered in the specification of the set of potential sources of influence.

An agent can perform a subset of scheduled actions if it receives permission to do so and satisfies its internal and "social" constraints. Such constraints are related

to its interactions with others. The permission an agent must receive to execute a transformation on the environment is called a *time-token*. To access the *time-token* in order to progress to its next state, the agent "asks" for it. The policy for *time-token* distribution is a matter of group representation as we shall see in the next chapter (see Section 6.7.6).

Whenever an agent is going to ask for the *time-token* it accesses its POR and changes its scheduler so that waiting processes may have their turn as well. There is no central clock in the environment, but each agent has its own version of global clock and whatever the policy used for *time-token* distribution the current global time will always be the one associated to those agents currently executing their actions. Such currently global time is uniquely represented in the special agent which represents a group of agents (see Section 5.6.2).

5.6 Group of Ecoagents

5.6.1 Environmental Agent

It is commonly believed that the behaviour of individuals may be influenced, by some special property of the set to which they belong, after a period of time much longer than the period of their localised actions. This raises another important issue: there may be interactions between properties of a group of agents and the agents themselves. The outcome of such interaction may produce a different behaviour in each agent of the group. This means that a group of agents should be treated as an special agent with special features responsible for the influences upon its components.

An approach similar to this, at the level of architecture, is defined as *coordination agent* [Kreifelts & vonMartial 91]. The purpose of the authors is to take advantage of the fact that reasoning about the coordination of other agents demands few resources. Although in many situations this coordination information can also be distributed to the agents, for domains such as ecosystems the whole environment often appears to have its own properties and behaves like an agent (*e.g.* a forest can be seen as a big leaf). Thus, it is almost fundamental to have an agent as the mediator for the overall system behaviour. This does not mean, however, that such an agent will care

about every kind of message exchanged between agents. It controls only the priority of actions according to their level of time scale, and also coordinates events that may cause change in the environment. I call such agent as *envagent* (which stands for environmental agent).

5.6.2 Knowledge Associated to an *Envagent*

An *envagent* has some knowledge about the agents of which it is composed. This does not need to be complete knowledge about the properties of agents but rather a description of the relationship between agents. This includes some notion of causation between events, processes, and actions of agents. It is important not to control these processes in themselves but to decide which agents should proceed with their behaviour when an important event happens. For instance, the growth process of a tree may not proceed within the flow of time until other agents, which it has some relation with, have finished their processing for the previous time. Furthermore, an *envagent* must be able to control the entering and leaving of ecoagents. For instance, when *bug_demon* is attacked by *John* it may leave the *envagent*, but may get in later.

Some possible attributes of an *envagent* E are:

- name and class of the environment,
- number of agents, their identification and class (possibly their location within the environment),
- dependency net or classes of influences among agents
- ("compound") attributes are the result of the composition (sum, product, etc.) of the attributes of its components (may be) plus some special property of the group in itself.

  Example: Biomass of a group of n trees (or animals) $B_1 + \ldots + B_n$
- geometrical shape (possibly sub-divided into zones, sectors etc.) with the identification and location of agents within it.
- current global time (local to the environment in relation to composition of envagents.)

It may be convenient to consider the subdivisions of an *envagent*'s topology as *sub-envagents*, each one responsible for coordinating the agents associated with it. For the sake of simplicity we do not consider this possibility in this work.

In order to obtain the value of one of its attributes, say A, E sends a message to its components requesting the value of their attributes associated to A. Along with such features there should exist the usual mechanisms for:

- coordinating the entrance and departure of agents,
- coordinating the access to the *time-token* as well as the order in which the action of external events should take place.
- coordinating events that change state of agents of the group
- answering questions about which agents are currently member of the group and how many agents are there interacting.

The coordination of entrance and departure of agents is important because it is in these two events that agents are informed about possible sources of influence. That is why a dependency net is an attribute of the *envagent*. Figure 5.3 shows an example of a net of potential influences involving three classes of *ecoagents*, where a directed edge indicates that the class of the starting node affects the class of the ending node. The double directed edges indicate that there are two affecting relation which may not necessarily be the same in both directions.

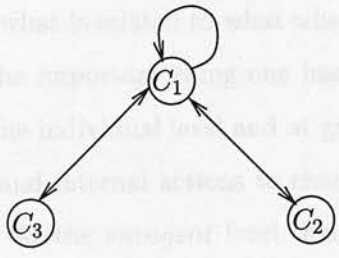

Figure 5.3: Example of net of potential influence among classes C_1, C_2 and C_3.

Such information is composed every time a new *ecoagent* enters into the environment. The agent sends the information about its class and the classes which are sources potential influence. In this way whenever an agent is

- introduced, we just search for a class in the net and we get the groups of agents should be notified about its entrance.

- moved out, we again just search for its class and get the groups which should be notified.

We can then define an *envagent* as a special ecoagent which coordinates a group of ecoagents. Formally it is defined identically to **Definition 5.6**. However, its special coordinating role means that in practice it is used in a restricted way. For example, its attributes are normally a function of the attributes of its members, its sensing and assimilating processes compose the behaviour of the group. Its main main functions are to handle: the location of agents within the structure of the environment it represents; the dependency net or classes of influences among agents; the scheduling of *time–token* which synchronise the actions of its members (events, entrance, etc.).

5.6.3 Similarities Between *Envagent* and *Ecoagent*

Given that we have a theory of time granularity and now a simple theory of ecoagents, is there any pattern of interaction between temporal and structural scales? In other words, is there any similarity between the behaviour of an *ecoagent* and an *envagent*? In what follows we have a discussion on this matter and I propose an answer.

One natural intuition is that an agent can be seen as a group in which the various processes of it are small agents, and the overall behaviour of them is related to the behaviour of the agent. But what is related to what when one tries to map from agent to group and vice-versa? The important thing one has to note is how change takes place at both levels (i.e. at the individual level and at group level). At the agent level this is related to attributes and internal actions to change it, which I shall represent by $ecoagent(Actions, Atts)$. At the *envagent* level this is related to agents' external actions and agents' state, which I shall represent by $envagent(Ecoagents, GAtt)$.

At the individual level, we saw that change is a composition of sensing, assimilating and executing changes based on the resource information acquired. At group level, it does not make sense to talk in terms of sensing and assimilating, although this is distributed among the members of the group. But we may allow the whole group to have observer

processes for each group attribute results from composing agents' attributes. I use the observer process here because it is a special kind of sensing message which does not absorb information but is simply able to monitor. Instead of resource information, it is called *group resource*.

After this, in the same way that an agent uses information resource to assimilate it into attribute change factor we may refer to a "group change factor". For compound properties like the example given above, this is just a straightforward composition of the values of the attributes of the group members. However, we may wish to associate certain properties which are not a composition of the members of the group. Properties of this nature I shall call *group properties* to mean that they can not be found in any individual.

This gives us enough information to see some similarity between agent and group, at least from a structural point of view. The important thing to note is the computation of a state of an agent and the group at a certain time. This is depicted in Figure 5.4, where observer processes (OP) at group level corresponds to sensing (S) resources at agent level; group resource (GR) corresponds to resource information for agents (Res); their assimilation processes have the same purpose but one produces a sort of group change factor, and the other attribute change factor. Note that Group Property may depend or not of the previous state of each ecoagent.

How can this bring about any relation between temporal and structural scales? To simulate the behaviour of a group in time, the actions of agents need to be synchronised in time. If we look at process ordering within an agent and try to map it to the group level, then we end up with a structure similar to:

$$\langle (T_1, \{A_1; \ldots; A_i\}), \ldots, (T_n, \{A_l; \ldots; A_m\}) \rangle,$$

where each T_i is a global time stamp, and each associated A_j is the identification of an agent whose next action is expected to take effect at time T_i. Here, instead of processes we have agents because at the group level it is not relevant to "know" which process of the agent should be executed for it is encapsulated in the agent's overall behaviour. An *envagent* does not generate the time stamps of this scheduler, it receives them from the components of the group and will perform temporal reasoning to order their request

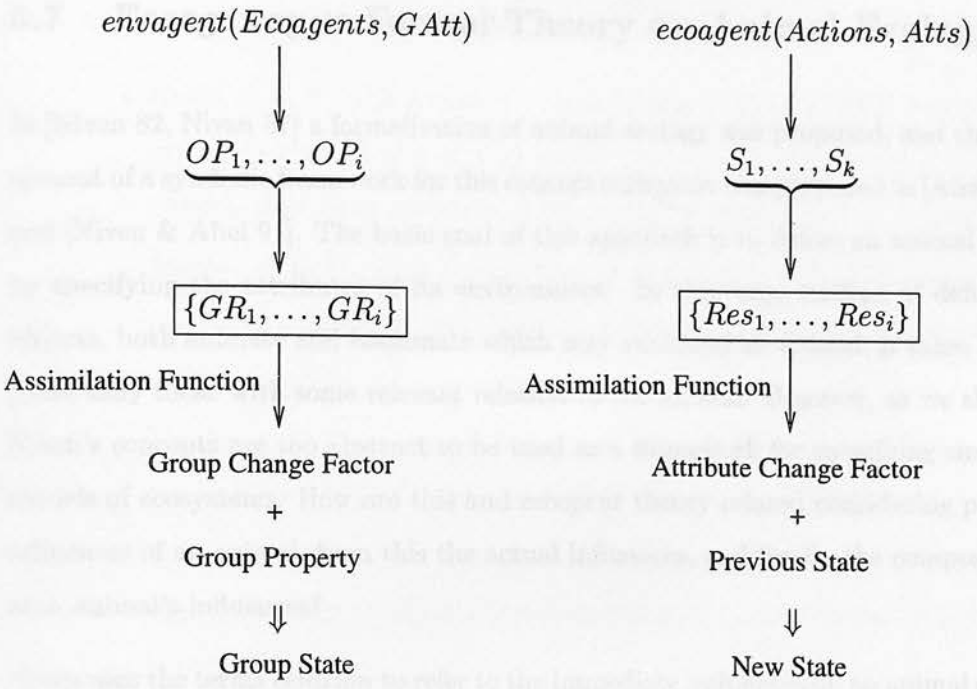

Figure 5.4: Similarities between Group and individuals.

according to their scale of time.

Thus, an *envagent* has a temporal granular scheduler similar to any *ecoagent*. This does not mean, however, that the control of the overall system is centralised, for an *envagent* does not interfere in the interaction among agents. An *envagent* only controls which agent is expected to change its state at a given time, but the result of their actions, the messages they might need to exchange in order to perform the action are left for them to resolve.

The gain with this comparison is that we may use the same framework to defining both *ecoagent* and *envagent*. They have similar internal representation, similar way to coordinate actions related to their behaviour. The differences will rely on the interpretation of their attributes and their model of execution. In Chapter 6 we describe how this can be done, and in Chapter 7 we show an application of it.

5.7 *Ecoagency* × Formal Theory on Animal Ecology

In [Niven 82, Niven 87] a formalisation of animal ecology was proposed, and the development of a symbolic framework for this conceptualisation was proposed in [Abel & Niven 90] and [Niven & Abel 91]. The basic goal of this approach is to define an animal ecology by specifying the attributes of its environment. In this way, instead of defining all objects, both animate and inanimate which may surround an animal, it takes into account only those with some relevant relation to the animal. However, as we shall see, Niven's concepts are too abstract to be used as a framework for specifying simulation models of ecosystems. How are this and *ecoagent* theory related considering potential influences of an animal, from this the actual influences, and finally the composition of each animal's influences?

Niven uses the terms *centrum* to refer to the immediate influences on an animal and *web* to refer to the network of interactions via other environmental entities. I will show that they have correspondent structures in the *Ecoagency* approach. However, *Ecoagency* is not a new theory for ecology, but rather a framework for developing simulation models of ecosystems. In this section we shall see that such a framework has a correspondence with Niven's formal theory for animal ecology. I will show that *ecoagent* theory has the advantage of being conceived to represent specifications that can be executed in a distributed architecture. The theory to be compared with *ecoagent* approach will be referred to as **Niven**, after its author. For simplicity of notation, the **NatureTime** language will be used for expressing temporal statements rather than Niven's original syntax.

Potential and Actual Influence over Animals

Niven - Uses a sort notation for animal and a pure number to represent either the expectation of life of an animal or the probability it has an offspring at an specific time; $Off(X,Y)$ @ T means that Y *is an offspring of* X *at time* T, and $\xi(X,Y)$ @ T means that X *provokes some immediately physical, physiological or behavioural response or change in the spatial position of* Y *at time* T, *because of a close physical proximity.* These are used to define the influences over a subject

animal, referred as **Niven-I**, as follows.

- $Res(r, a)$ @ T - *r is a resource for a at time t if $\xi(r, a)$ @ T holds and the expectation of life of a at T is greater than it was at α before T. If r is an animal, then its expectation of life decreases from α before T until T.*
- $Mat(m, a)$ @ T - *m is a mate of a at time t if $\xi(m, a)$ @ T holds and the probability that both have the same offspring at time $T + \beta$ is greater than zero.*
- $Pred(p, a)$ @ T - *p is predator of a at time T if $\xi(p, a)$ @ T holds and the expectation of life of a decreases while of p increases from α before T.*
- $Mal(c, a)$ @ T - *c is a malentity for a at time T if $\xi(c, a)$ @ T holds and the expectation of life of a decreases while of c may decrease or not from α before T.*

Ecoagent-PI - Is defined in the set of potential influence and definitions of function of influence (Section 5.3.1). More specific than **Niven-I**, the semantics says that these are potential sources of influence and also under which circumstances their instances become actual influences.

There are two advantages in the representation of influences over an animal by using this. First, potential influences must be explicitly defined in terms of the attributes and processes of a particular model, while **Niven-I** is highly general, depending on abstract (and undefined) concepts like expectation of life in order to determine potential influences. Second, there can be more environmental relations which **Niven-I** is not able to represent. This means that not only the model of an animal can be defined in the same way but other entities of the environment, living and non-living organisms.

The *Centrum* of an Animal

Niven-C is the set of things (or objects) which directly affects the animal. It is composed of resource, mates, predators and malentities. The *centrum* C of a subject animal a at a given time t is the set of all objects x which satisfy the condition of **Niven-I**.

$$C_t = \{x \mid Res(x,a) \ @ \ t \vee Mat(x,a) \ @ \ t \vee Pred(x,a) \ @ \ t \vee Mal(x,a) \ @ \ t\}$$

In [Niven & Abel 91], Niven and Abel suggest that objects classified in these categories should be understood as what it is used in every-English as potential resources, potential mates, etc.. What they seem to mean is that the definitions in **Niven-I** are potential influences and the actual influences depend on which instances of them are in the environment. However, there is no clue on how this can be obtained from their specifications.

Ecoagent-LE Is defined by an agent's Local Environment (LE) (see Section 5.3.2). This structure is distinct from **Ecoagent-PI**, although it is derived from it. This clear distinction helps us to envisage which mechanisms are involved in the derivation of the former from the latter. This is not the case in **Niven-I** and **Niven-C**.

The *Web* of an Animal

Niven-W Are those things which indirectly affect the animal, and they are called *modifiers*. According to the order of the modifier it may belong to the *centrum* (zero order), or it is a modifier of lower order modifiers. I will write $mod_n^+(w,a,M) \ @ \ T$ to mean that the set of positive modifiers at level n of web w of an animal a is M at time T, analogously $mod_n^-/3$ will represent the negative modifiers. These modifiers are defined as follows.

$mod_{n+1}^+(w,a,M) \ @ \ T \Longleftarrow$

$M = \{x$ such that x modifies positively at level n either a or $w \ @ \ T\}$

$mod_{n+1}^-(w,a,M) \ @ \ T \Longleftarrow$

$M = \{x$ such that x modifies negatively at level n either a or $w \ @ \ T\}$

A modifier is considered to be positive or negative depending on whether its presence or absence causes the lower order object to appear in the system.

Ecoagent-W As *centrum* and *web* are supposed to be actual instances of influences, according to *ecoagency* framework they can only be represented in the local environment. In the definition of PI of an *ecoagent* there is no such a notion of

negative or positive influence, but this can be easily obtained from the function of influence associated with them.

If we were to represent such modifiers in the local environment of an agent, then it would be equivalent to representing the whole environment in a single model. This means that every agent would have complete knowledge about the world. However, the web of an animal is distributed in the representation of LE of group of agents, i.e. $Web(A)$ is the union of the local environments of the the elements of A's local environment along with their web, as shown below.

Suppose *ecoagent* A has $\{B_1, \ldots, B_k\}$ as its local environment at a given time t and there are n agents in the whole ecosystem. The modifiers of each B_i, according to *ecoagent*, are in its local environment. Let $lenv_n(X)$ be the local environment of agent X at level n and $web_n(X)$ be the web of X at level n. Then we may recursively define the web of an agent A as follows.

$web_n(X) = \{y \mid y \neq X$ and $\forall z \in lenv_n(X)$ either $y \in lenv_{n+1}(z)$ or $y \in web_{n+1}(z)\}$

From this we have that $web_0(A) = lenv_0(B_1) \cup \ldots \cup lenv_0(B_k) \cup \bigcup_{i=1}^{k} web_1(B_i)$. Each $web_1(B_i)$ can be defined as $lenv_1(lenv_0(B_i)) \cup \bigcup_{i=1}^{k} web_2(lenv_1(B_i))$, by defining modifiers at each level n in terms of lower levels and so on. As each agent has its own model of representation and there are only n agents in the whole environment, then the complete web of A is distributed in the web of its local environment.

This view is depicted in Figure 5.5, where $A = a$ and its local environment is $\{b, c, d, e\}$. The arrows denote direct influences from the agent to the subject at the pointed end of the arrow. But those elements of a's local environment can be modifiers among them.

These concepts of *Ecoagency* are more specific and directed toward modelling than Niven's formalisation, but we may obtain similar notions of *web* and *centrum* via an *ecoagent*'s local environment.

The positive side effect of *Ecoagency* representation is more evident if we think in terms of execution or simulation of an agent's behaviour. The concept of potential

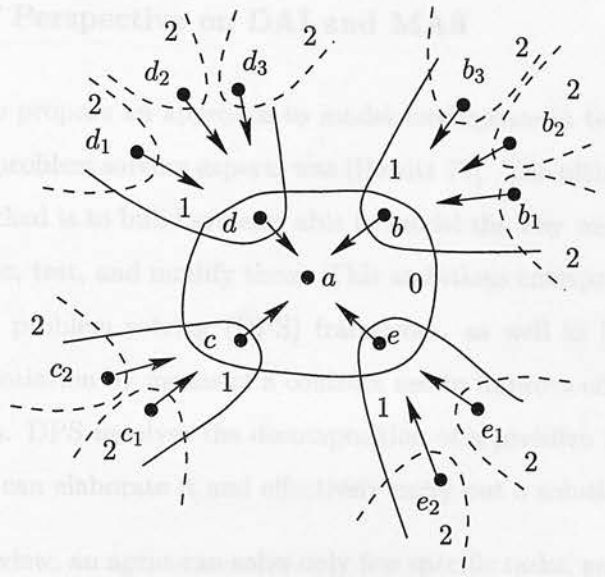

Figure 5.5: A net of modifiers of agent a. $web_0(a) = \{b, c, d, e\}$, $web_1(a) = \{b_1, b_2, b_3, c_1, c_2, d_1, d_2, d_3, e_1, e_2\}$.

influence (PI) can be seen as non temporal meta-knowledge which helps us to reduce the computation of searching for temporally actual instances every time the state of an agent A changes. The use of a general temporally dependent representation for influences over A, as Niven's approach does (or just using **NatureTime**), requires the knowledge, of all agents to be put in contact with A. This would impose a strong determinism on ecosystem modelling task. The **Ecoagent-PI** is an abstraction which allows us to reason only about actual instances of PI.

5.8 Ecoagent Theory in the Context of DAI

We now place the ideas presented in previous sections in the context of distributed artificial intelligence (DAI). First, we present a brief overview of DAI, MAS evolution, then we shall see the propositions to classify agents, and formal languages to program agents. During the presentation we shall compare the concepts presented with *ecoagent* theory.

5.8.1 A Brief Perspective on DAI and MAS

One of the first to propose an approach to model intelligence in terms of a society of knowledge-based problem solving experts was [Hewitt 77]. The ultimate aim that such an approach launched is to build systems able to model the way we construct our theories, communicate, test, and modify them. This ambitious enterprise found problems on its distributed problem solving (DPS) framework, as well as [Davis & Smith 83] who proposed negotiation by means of a contract net to improve efficiency in problem solving techniques. DPS involves the decomposition of a problem in such a way that a group of agents can elaborate it and effectively carry out a solution [Minsky 86].

According to this view, an agent can solve only few specific tasks, and to solve complex tasks we must sub-divide it into sub-tasks and allocate them to the agents of the society which are able to solve each one of them. This idea along with the interest of exploring concurrency and distribution in Artificial Intelligence computations, at many levels yielded the area of research called Distributed Artificial Intelligence (DAI). The bottleneck of DPS is that agents or informations can not be introduced or retracted, and the system could not adapt itself to this new situation. Then the idea of open systems came about and [Hewitt 88] addressed the main points associated to such kind of systems: concurrency, asynchrony, decentralised control, inconsistent (may be incomplete) information, autonomy, and continuous operation.

An autonomous agent has its local (or individual) knowledge, and is able to act on its environment. According with the changes it produces and the reaction of the environment, it may change its local knowledge as [Kreifelts & vonMartial 91] pointed out. A composition of such agents was then called Multi-Agent Systems (MAS).

The problem of what should constitute the features of an agent in the context of KBS is addressed in [Sichman & Demazeau 92], and this is the most common in DAI literature. An agent emerges from the KBS when it is introduced *social reasoning*, *perception*, *control*, *actions*, and *communication*, as shown in Figure 5.6. In fact, the knowledge base is partitioned into knowledge acquired by communication exchanged and perceived in the environment.

The important points we should consider about MAS from the KBS context are:

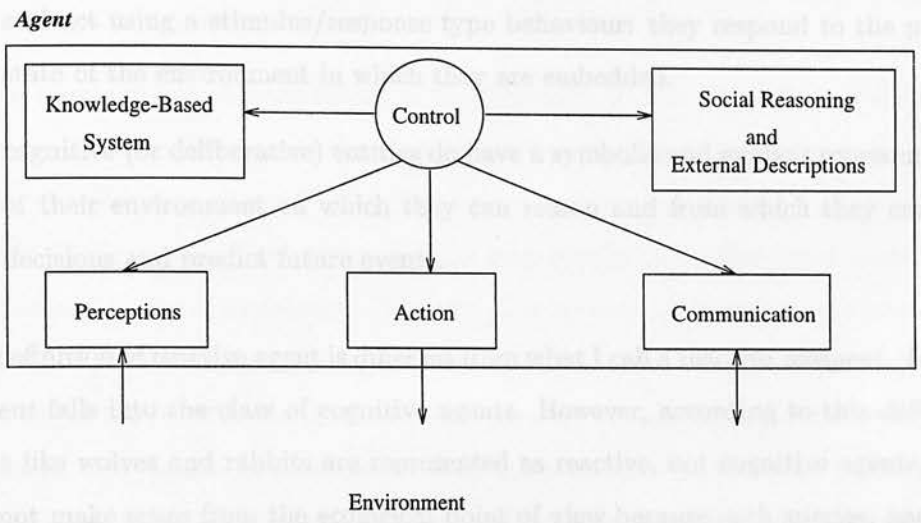

Figure 5.6: Extracted from [Sichman & Demazeau92], a cognitive concept of agent.

- the description of possible connections and interactions between knowledge bases.
- the temporal properties which are successively satisfied during an agent's reasoning.
- the static or dynamic properties of a group of agents, and their influence in the temporal features of the agents and their reasoning process.

Along with this there is the problem of what is an agent, and which classes of agents can we envisage for our purpose of developing MAS applications. I discuss this in the next sections.

5.8.2 How Agents Have been Classified

[Wooldridge & Jennings 95] pointed out the current view of agent where two branches have been the main directions of research in this field. The first is called a Weak Notion of agency where the fundamental properties are autonomy, social ability, reactivity and pro-activeness. The second is called a Stronger Notion of agency which is based on mentalistic view of agents. [Ferber 96] have also suggested that the computational entities (or objects) of Distributed AI can be divided into two classes which are related to these weak and strong notions of agency. These are:

- reactive entities are those which do not have representation of their environment

and act using a stimulus/response type behaviour: they respond to the present state of the environment in which they are embedded.

- cognitive (or deliberative) entities do have a symbolic and explicit representation of their environment on which they can reason and from which they can take decisions and predict future events.

This definition of reactive agent is different from what I call a reactive *ecoagent*. Indeed, *ecoagent* falls into the class of cognitive agents. However, according to this definition agents like wolves and rabbits are represented as reactive, not cognitive agents. This does not make sense from the ecological point of view because such species, and even some plants seems to have their own "representation" of space-time [Levin 92]. They can not also be included within the category of cognitive agents because they not necessarily predict future events. The limitation of this division above is that it assumes that having a symbolic representation of the world means the ability to make prediction (or deduction) about it.

[Goodwin 92] proposes that we should describe an agent system in terms of the agent itself (its internal state, mechanisms for sensing and acting), the environment and the tasks the agent can perform on such environment. There is a formalisation of concepts such as successful, capable, perceptive, reactive, reflexive, perceptive. An agent does not need to reason about others in order to be successful, but just observe their behaviour. Godwin's approach uses functions to estimate the external state that agents have. *Ecoagency* approach does not require it as necessary, although they could be specified in the way agents sense information resource and assimilate them.

[Shoham 93] proposes a concept of agents based on mental states which can be ascribed to a machine or computer program. The state of an agent is defined in terms of belief, choice, commitment, etc.. Again, *ecoagent* approach gives a more general and abstract description of an agent and its interaction with the environment. One may want to associate mental states to internal attributes, but then one it has to add special mechanisms for reasoning about beliefs and other usual mental attributes. I chose to take the view of assimilation function because it embodies all these concepts, and the cases where mental states are being considered can be treated as properties of active

ecoagents.

Most of these work (and many others in the literature) do not have a class of agent such as the *passive ecoagent* proposed in this thesis, but they treat it simply as an object. The closest idea that it could be useful to treat such entity as agent was proposed in [Luck & d'Inverno 95]. They argue that one may ascribe to a coffee-cup a mental state but it would have no usefulness. However, if it contains a liquid it will serve some purpose, and thus will act as an agent. It could be used for other purpose and then it would be an agent. If it is empty and in a purposeless position then it could not be considered as an agent. In other words, they consider that "an entity does not need to be intelligent for it be an agent". But this view differs from ours when they consider such entities as a coffee-cup as transient, i.e. an object which becomes an agent at some time, may subsequently revert to being an object.

However, this does not make sense in the *ecoagent* theory because active *ecoagents* do not become reactive, reactive do not become passive! The example given of a cup being an agent when it is serving a purpose (full) and an object when it is empty, does not justify the generalisation of the concept. For a cup, even empty can be seen (in our view of agency) as a passive agent with empty resource. What it is going to be made of it is another matter, but it is still passive.

According to this transient concept a robot which is without power, and so cannot use its actuator is not very helpful. Thus it would not be an agent but an object. For a dead rabbit is still a rabbit, but which is unable to respond to certain messages. Would it be considered as passive? Well, according to Luck and d'Inverno yes (object in their case), but nothing is said in their theory that it could not be able to become a reactive rabbit *ecoagent*. However this is not plausible and it is biologically impossible. Therefore the transient feature for an agency does not seem to be a good one as it was stated. I agree that a robot which has power supply restored may become an agent again, but in the sense of *ecoagency*.

5.8.3 Agent's Potential Influence and Environment

To our knowledge almost no work in AI or DAI has proposed deeper investigations into the nature of knowledge associated with potential sources of influence. This describe not just potential relations, but rather a scheme of meta-rules to derive relations only when certain conditions hold. The terms $value(att, agent(class))$ or $absorbed(amount(X, Att), agent(Class))$ are interpreted in a way that takes time into account. The scale of time is the one defined for the process which is affected by these components. McCarthy and Hayes, [McCarthy & Hayes 81], call such time dependent functions as *fluents*.

A first work to propose a model of an agent which considers the local environment as an important aspect was [Hewitt 77]. Hewitt Actor's model, in general terms, defines an actor in two aspects related to the behaviour of it: the actions it should perform when it receives a message and its ACQUAINTANCES or a finite collection of other actors it knows about. By "knows about", he meant the actor knows the nature of the relation between it and each member of its ACQUAINTANCES. Such a relation may be asymmetric in the sense that the reverse of the relation may not be the same.

The local environment of an *ecoagent* captures this idea. But we provide a framework and a mechanism to change such a local environment according to the interaction between the agent and the environment. This is the set of potential influences.

In [Ferber & Jacopin 91] an expression similar to our *ecoagent* is used to denote the basic entity of a framework called Eco Problem Solving (EPS). An eco–agent is actor-based [Agha 86] and has local knowledge where the agent knows: whether or not a goal has been satisfied, the current dependencies between it and other agents which can be master or slave, and agents called jailers which prevent others from acting. However, EPS is a framework for distributed problem solving which has the limitation that the full tasks should be previously specified to the agents, and so agents cannot be added to a society or retracted from it. The local knowledge concept does not capture the quality of the possible relationships between agents.

[Ferber 96] proposes as a model for multi-agent systems a quadruple composed of agents,objects,environment,communications. Agents are defined according to their

ability to perceive specific kinds of communications, have skills in performing actions, deliberation model (if there is one), ways to relate perceptions to actions; objects are all represented passive entities that do not react to stimuli (*e.g.* furniture); environment is the topological space where the agents are located; communications are categories of communication (sound/etc.).

This definition does not make any distinction between the classes of agents according to their perception of certain actions. Also to treat passive entities as "objects" does not help in the case of complex passive entities which offer resources to others such as soil, river, etc.The view of environment is very restricted because it does not provide abstract information about the group of agents.

5.8.4 Environment or Group of Agents

In [Goodwin 92], the model of the environment is non-deterministic and time is represented in a discrete framework. The smallest time interval is associated to an integer in order to keep track of the history (or Chronicle [McDermot 82]) of actions and change of the state of an agent. The size of such a smallest interval called quanta determines the level of accuracy one wants to approximate discrete to continuous time.

The environment is schematically represented by "laws of nature". This should describe the valid combinations of states of the world by means of probability distribution on how things might possibly change.

[Fisher & Keane 95] also propose that the properties of a group objects (or agents) should include the ability to send a message to a group, add/remove an object to/from a group, ascertain whether a certain object is member of a group, and construct a new group. We can see the construction of group in *ecoagent* theory as gathering agents according to their influence relationships, but the *envagent* is not the one which builds the group. It only keeps track of the classes of groups which may affect each other.

5.8.5 Languages for Programming Agents

Once we we have defined what our agent should be composed of and the way in which group of agents should be represented, then a natural concern is how we are going to

program a MAS application. The number of languages which have appeared is great and it is very difficult to find one which is a standard or a paradigm of programming language for agents. There are many languages for handling inter-agent communication such as KQML [Cohen & Levesque 95], and most of these have been used for cooperation and coordination of actions of agents, *e.g.* COOL [Barbuceanu & Fox 95]. But most of them do not provide mechanisms for execution for they are not concerned with simulation of agents.

One of the first formal logic-based language for MAS which meant to be executable was proposed in [Fisher & Barringer 91]: concurrent METATEM processes (CMP) is a high level language for modelling distributed concurrent systems. The first aspect of CMP is the executable temporal logic of METATEM, [Barringer *et al.* 89], in which every formula is in the form $P \Rightarrow F$, that is, if the properties P hold in the past then the properties F will hold in the future. The second, as a framework for concurrent systems, processes specified in this language communicate to each other through message-passing like in the Actor Systems.

Later on, [Fisher 94a] defined this approach as a combination of an executable temporal logic and concurrent object-model, where each object (or agent) is specified in this logic, and the interaction with other agents is done by broadcast message-passing. No formal theory was proposed to address hierarchies of communication among agents. No imposition is made on the kinds of messages agents are allowed to send or receive. This logic has been advocated as a framework to develop DAI applications in [Fisher & Wooldridge 94] where its applicability to specifying protocols for cooperative action is presented.

Although CMP is logic-based and has meta-level capabilities, it offers no framework to describe and reason about the time varying properties of MAS. The language is also too general in relation to kinds of messages allowed. Furthermore, its heavy logical notation and lack of structures to organise the components of an agent would not help in creating discipline in the modelling process.

Another formal language was proposed in [Shoham 93], called AGENT0 which is Shoham's *Agent-Oriented Programming* (AOP) paradigm. This is based on a view of

computation which depends of a society of agents (or objects). The language of AGENT0 is a simple linear temporal logic where predicate and modal operators are "time stamped". In the same way as CMP it has no structure for representing actions interacting at many scales of time.

The fact that we can specify the behaviour of agents by using languages which can be directly executed, such as CMP or AGENT0, does not imply that it is always helpful to offer a language with general features. Although concurrent METATEM has been proved to be useful for modelling reactive systems, in general, [Fisher 93], its straight use relies on many skills of the programmer, from modelling knowledge to the level of coordination of groups of agents. Let's assume that such languages could be extended to satisfy the problems addressed in Section 2.4 (there is no reason to believe they could not). Except from minor differences on syntax notation, we would end up representing simulation models of ecosystems in a scheme very similar to GSCS (see Section 5.4).

The level of agent programming that seems to be most useful, as far as *ecological modelling* is concerned is the how agents influence each others (potential influences), what should they do in the presence of others (functions of influence plus appropriate message exchange), what are the actions on agent's attribute (functions to compute actions $\mathcal{A}$). We have argued that there are correspondences between these features and the ones presented above. It would be interesting to investigate in more detail how this "functional" perspective of an agent's attribute and processes is related to such mental or deliberative views.

5.9 Summary

In this chapter we achieved half of one of the aims of this work (see Section 1.5) related to the development of an agent based framework for representing simulation models of ecosystems with their own view of time and local environment. The theory presented here is meant to be more a guideline to develop *ecoagent* based simulation rather than the framework in itself. An example of a possible way to implement it will be seen in next chapter. We now summarise the main point related to *ecoagency*.

- Agents are grouped in classes according to their behaviour and level of awareness

about themselves. We use the term *ecoagent* in this sense.

- An ecoagent manifests its action upon the environment by means of messages it sends to the object of its action.
- Ecoagents can be divided into three main categories:
  - *Passive* - it never sends requests or queries to other agents, but always offers something (resource) to others.
  - *Reactive* - it may send requests, queries to the environment, and it may send some orders to agents of its own class.
  - *Active* - it may send any kind of message as those shown in Table 2.1, but it is more aware about itself (actions, goals, beliefs, etc.) than the others.
- An *ecoagent* has
  - a set of potential sources of influences, where it is defined which classes may affect the agent in the case instances of them satisfy certain constraints,
  - local environment which represents actual instances of potential influences specified,
  - capability for sensing, interpreting and assimilating external information into meaningful information from the agent's perspective,
  - actions of agents can be related to the execution of general simulation clause schema which represent such actions. The execution of such actions is related to operations over the structures and use of functions mentioned above.
- An *ecoagent* has its own scale(s) of time for its process(es). Such scales are modular sets of a finite hierarchy of such sets. This is used by its scheduler of actions which may consider other restrictions like causal relationship among them.
- An *envagent* is a special *ecoagent* with skills to coordinate or synchronise the actions of a group. But it also holds important properties of the group such as:
  - number of agents, their identification and class (possibly their location within the environment),

- dependency net or classes of influences among agents
- ("compound") attributes are the result of the composition (sum, product, etc.) of the attributes of its components (may be) plus some special property of the group in itself.
- topological shape (possibly sub-divided into zones, sectors etc.) with the identification and location of agents within it.
- current global time (local to the environment in relation to composition of envagents.)

- *Ecoagency* is specific and directed toward modelling than Niven's formalisation, but we may obtain representation for *web* and *centrum* via an *ecoagent*'s local environment.
- We have argued that our framework raises an interesting research question: how *ecoagency* functional perspective of attributes, processes and functions is related to the usual notion of deliberative agents? To what extend such alleged correspondence could help us to delineate limits for computability of such theories?

Chapter 6

An Architecture for Ecoagent Systems

6.1 Introduction

In chapter 5 a theory of agents was presented from an ecological modelling point of view. We saw that the structures necessary for dealing with *general simulation clause schema* were needed for the *ecoagent* theory, but they were not detailed because they may be implemented in many different ways. In this chapter, I shall describe one way of implementing the concepts of *ecoagent* and *envagents*. As passive agents are a simple case of reactive, we shall not describe it here. In the end, I show that the *ecoagent* based simulation of ecosystems in this framework is free of deadlocks.

6.2 Basic Assumptions

6.2.1 Symbolic Architecture

The central computational idea behind any DAI program or language is to reduce the search space (within an agent) by distributing parts of it. Whenever an agent needs information about the world, and this information does not concern the agent itself, then the agent interacts with the environment. This is always the case no matter whether the architecture is symbolic or reactive or hybrid. As far as symbolic architectures in DAI are concerned the world (the external information of the agent) is symbolically represented, as well as the interactions happening. As an agent "feels"

the world through such interactions, then we may say that from the agent's point of view the environment is represented by the sets of messages the agent sends and receives. For example, in [Brazier *et al.* 95] it is suggested that an observation of the external world can be seen as an agent's specific request for information about the world through an information link.

6.2.2 Communication

As far as the process of communication between *ecoagents* is concerned, the following things are assumed in this work.

- The stream, or medium of communication, through which agents send and read messages is like a server entity to which every agent (including *envagents*) is a client (we could assume more servers as suggested in Section 7.6).
- An agent can only access messages sent to itself, to the group it belongs or to the global pool of external knowledge.
- The medium of communication is also used to put information about the external interface to the system, i.e. the set of external knowledge of each agent. Only the agent can destroy or update the contents of this external information and this may happen as a result of its interaction with the environment.
- The order in which messages arrive may be different to the order they are sent. This is also true for messages about external events since they are represented by means of messages sent to the *envagent*.
- Whenever an agent sends a message it is delivered, i.e. an ideal medium of communication is assumed.
- Whenever an agent is trying to read a message it will eventually succeed, but this may not necessarily mean the agent will be in a waiting condition all the time until the message arrives.

6.2.3 Atomicity of Computation within an Agent

To simplify the design, the level of atomicity of an agent's overall computation is limited to the level of message read from the environment. This means that an agent will read and process just one message at time. Only after processing a message it may read another one, and so on. A more sophisticated system might allow agents to read more than one message and process them in parallel. However, this introduces another level of complexity which we avoid in this thesis.

As the order in which the messages are received is non-deterministic, then the outcome of an agent's overall computation through the logical time should be similar to the one generated by another agent with real parallel processors. The only difference, as stated above, is that the latter kind of agent architecture would need a more detailed specification of its parallel processes accessing shared knowledge.

The implementation chosen to experiment with these concepts of agency is physically different from a full parallel architecture. However, it is legitimate to say the behaviour of ecoagents is concurrent because the non-determinism of message exchange and processing simulates concurrent processes on sequential machine. In fact, as [Frolund 96] suggests in concurrent object-oriented paradigm objects (or agents) have one thread of control, which means that only one method can be executed per time by object. It is proved that "the effect of internal concurrency is equivalent to executing the methods one at time".

6.2.4 Multi-Agent Coordination: A Pragmatic Solution

As we are interested in simulating the behaviour of individuals and groups of *ecoagents*, the only "computational resource" common to all agents is the permission to progress through the global logical time (or *time-token* as presented in Section 5.5). The coordination for agents to access the *time-token* is done by a special agent called the *envagent* (see Section 5.6.1).

A non-centralised approach for distribution of the *time-token* could be used, but such a policy is more useful in a domain where all agents are competing for few shared resources, which is not always the case of ecological simulation. Furthermore, this

would be a point-to-point message passing approach which has two main disadvantages. First an agent has to "know" about every other agent in the whole environment, even about those which its behaviour or internal states has no relation with. Second, to model an open system would be much harder because each agent has to be able to coordinate events not related to its own behaviour. This may lead us to a huge amount of replication of data.

6.2.5 Resource Acquisition and Interaction

An agent does not depend on the *time–token* to acquire the resource it needs to progress its internal state. An agent is able to exchange and negotiate directly the resources it needs with others as pointed out in Section 5.6.3. However, it does not mean it performs actions to change its state for it needs permission to do so. Also, an agent does not need to send messages to all individuals within the environment to acquire resources, but only to those which belong to its local environment.

The best way to understand this is to separate computations into two levels. At one level, there are the ones related to changes (possibly at many levels of time scale) on the attributes of the agents. This is the logical time at the *ecoagent* state level. At another level there are computations related to interactions between an agent and those members of its local environment in order to obtain information from them that it might need. This one is the architecture state level as depicted in Figure 6.1, where doubled arrows represent potential interactions at the architecture level, single arrows are functions of state transition $(\mathcal{A}_1, \mathcal{A}_2, \mathcal{A}_3)$ after permission being granted (dashed arrow), and the curved areas at *ecoagent* state level represent state boundaries.

Note that A_4 does not change its state. This may happens because the time scales of its actions have been specified to work at a coarser grain. It could also be that A_4 is non-aligned in time with the others agents.

6.3 Ecoagent Representation and Definition

The first thing to introduce is the external representation which is a sort of *interface definition* as proposed in [Fisher 94a, Fisher 94b]. In this approach, an agent interface

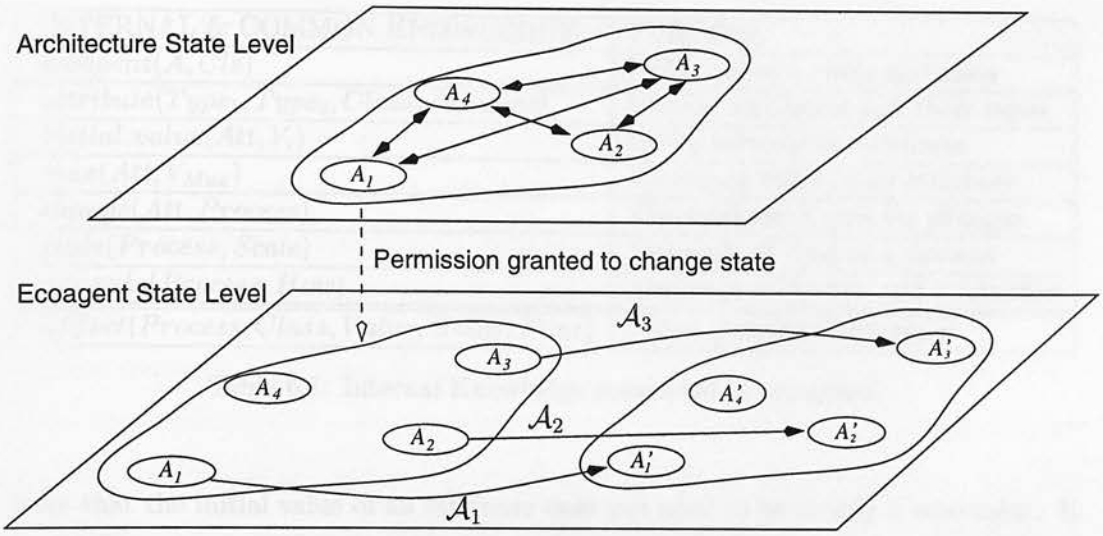

Figure 6.1: Architecture and Ecoagent State Levels.

is represented by its name, the messages it can recognise and messages it can produce. This is defined as follows.

Definition 6.1 (External Knowledge (or Interface)) *Let $Ag \in \mathcal{N}_{ag}$ be an agent identification name, $Cls \in \mathcal{N}_{cls}$ be the name of Ag's class, $Env \in \mathcal{N}_{ag}$ be the envagent for which Ag is a member, Msg_r is the list of recognizable messages, Msg_s is the list of messages Ag is allowed to send, and $atts(AttVals)$ is the set of external time independent properties (or attributes) of Ag. Then, the* External Knowledge (or Interface) *of ecoagent Ag is $ecoagent(Ag, Cls, Env, Msg_r, Msg_s, atts(AttVals))$.*

This definition is represented in the medium of communication (say Ω). An agent will write its interface on Ω by using a primitive *write_ek*/1 which takes as a parameter the term defined above. As part of the agent definition we have the specification of attributes, internal processes with their time scale and computational functions, the functions of influence over processes. This is called the *individual internal knowledge* of an agent. The set of definitions for potential sources of influence composes a sort of *Common Hierarchical Knowledge* (CHK) among classes of *ecoagents*. In what follows $Type_1$ can be either *external* or *internal* and $Type_2$ can be either *dynamic* or *static*, and the last row shows the CHK definition.

INTERNAL & COMMON KNOWLEDGE	PURPOSE
$ecoagent(A, Cls)$	Define agent's name and class
$attribute(Type_1, Type_2, Class, Attname)$	Declare attributes and their types
$initial\_value(Att, V_i)$	Initial value of an attribute
$max(Att, V_{Max})$	Maximum value of an attribute
$change(Att, Process)$	The attribute a process changes
$scale(Process, Scale)$	The scale of time of a process
$compute(Process, How)$	Associate a Process and a function
$affect(Process, Class, Value, Holds, Func)$	Define potential influences

Table 6.1: Internal Knowledge associated to *ecoagents*.

Note that the initial value of an attribute does not need to be simply a statement. It could be a rule which determines, for each instance of that class what are the initial values for its attributes, for instance $initial\_value(Att, Area, V) \Longleftarrow V\ is\ F(Area)$, where F represents a function which computes V according to constraints on the area. The other kind of knowledge is the mechanism for dealing with interaction or message exchange, but this should be transparent to the modeller.

By using this *ecoagency* theory, modellers write their models for representing individuals or groups of them by using the same notation as presented in Table 6.1. They also have to define the domain specific features of the model or the definitions of the functions to compute change and influences (the modeller can take advantage of a library of such functions). Then, we offer mechanisms of execution to make such models interact, no matter their level of time or structural granularity. Such mechanisms compose what we call an ecological agent-based simulation that we describe in the rest of this chapter.

6.4 Messages for Ecoagent Communication

6.4.1 Primitives of Communication

An agent A will always read messages from the stream of communication by using either a communication predicate $wait(msg(A, Msg))$ or $read(msg(A, Msg))$. The difference between both is that $wait/1$ keeps the agent waiting for any message sent to it, while $read/1$ fails if there is no message in the stream and the agent may try

another task, for instance answering unattended queries. The first kind of reading message will be used in cases where the agent's control is not allowed to do anything before receiving a specific message. For instance, when an agent asks permission to enter into an environment it can only start interacting with others after receiving the permission for entrance. Other versions of $read/1$ and $wait/1$ are also used for reading external information without removing it from the medium of communication.

To send messages to an agent B an agent A will use $send(msg(B, Msg))$, for point-to-point communication to a limited number of agents. For a group of agents it will use $multicast(Group, Msg)$, and $broadcast(Msg)$ to all agents waiting for Msg. But this is only allowed in situations where the whole environment has to be informed about something. Whenever a message is sent to an agent it will eventually read it, i.e. I assume that there are no faults in the process of communication due to problems of transmission.

The informal meaning associated to sending a message is that an agent *writes a message in the stream of communication* to which it is linked. On the other hand, reading a message means that the agent *extracts a message from such a stream.* But an agent may read external information from it without taking it off. These concepts are formalised in Section 6.5.1.

6.4.2 A Simple Communication Language

The language for agent communication is composed of two sets. The first is $\mathcal{M}_{out}$ to represent the set of messages an agent is allowed to send. The second, $\mathcal{M}_{in}$ is the set of messages an agent is allowed to receive (or that it "understands"). The set of messages using $send/1$, $broadcast/1$ and $multicast/2$ taking the correct format of the envelopes into account compose the signature $\Sigma_{\mathcal{M}_{out}}$. Analogously, for $read/1$ and $wait/1$ we have $\Sigma_{\mathcal{M}_{in}}$. A message contents is defined, in BNF style notation, as follows.

$\langle Msg \rangle$ ::= $msg(\langle Agent \rangle, \langle MsgEnvelop \rangle) \mid msg(Msg)$

$\langle Agent \rangle$ ::= agent name identification

$\langle MsgEnvelop \rangle$ ::= $\langle MsgType \rangle(\langle Agent \rangle, \langle MsgContent \rangle)$

$\langle MsgType \rangle$ ::= $query \mid request \mid inform \mid answer \mid demand \mid$

$offer$ | $deny$ | $command$ | $reject$ | $accept$ |

$propose$ | $report$ | $suggest$ ...

$\langle MsgContent \rangle$ ::= $entrance(\langle AgCls \rangle, \langle ClsInfs \rangle, \langle Location \rangle, \langle IField \rangle)$ |

$entrance(\langle Perloc \rangle, \langle CurrentTime \rangle, \langle IndPotInfs \rangle)$ |

$entrance(confirmed)$ | $entrance(not\_possible)$

$token(\langle NextTime \rangle)$ | $token(\langle Perloc \rangle, \langle NextTime \rangle)$ |

$no\_influence$ | $inf(\langle InfContent \rangle, \langle AgCls \rangle, \langle AgState \rangle)$ |

$rel(\langle AgCls \rangle, \langle Rel \rangle, \langle Value \rangle)$ | $amount(\langle Value \rangle, \langle Resource \rangle)$ |

$departure(...)$

$\langle IndPotInfs \rangle$ ::= individuals who are potential influences

$\langle InfContent \rangle$ ::= new_agent | $value(\langle Att \rangle$ @ $\langle TimeMoment \rangle)$

$progress(value(\langle Att \rangle))$ @ $\langle TimeInterval \rangle$ |

OTHER POSSIBLE ACTIONS TO PERFORM

$\langle AgState \rangle$ ::= $state(\langle Time \rangle, \langle Atts \rangle)$ | $\langle StateHistory \rangle$ | im_in |

OTHER POSSIBLE STATES, *e.g.* *angry*, *happy*, *armed*, *hungry*, etc.

$\langle Location \rangle$ ::= agent's location (geometrical, structural, etc.)

$\langle IField \rangle$:= agent's ratio of influence

$\langle Perloc \rangle$ ::= $denied$ | $allowed$ | $forbidden$ | $suggested$ | $accepted$ | ...

Note that there are many possibilities of combining the components of an envelope. In this work only a limited set of combinations is considered. Content which obeys the syntax rules above is called a *well formed message envelope* (WFME). As pointed out in Section 5.2 not every agent is able to recognise all kinds of message. However, we may consider some of them according to the class of *ecoagent*.

6.4.3 Messages Allowed to Reactive Agents

In Table 6.2 Cls is the class of the agent who sends the message, $CInfs$ is the classes of potential influences of the agent, Loc is the location of the agent within the environment, $IField$ is an agent's field of influence, $Reason$ is an explanation for a certain action, Env is the *envagent* to which a message is sent, T is the next time of which the agent is going to progress, Λ_0 is the initial state of the sender, Λ_{hist} can be either

the value or set of values of an attribute at the last time as an answer to a sensing message. Requests are sent only to an *envagent*, queries are sent only to agents which are members of its local environment.

	MESSAGE TO SEND	PURPOSE
Request	$entrance(Cls, CInfs)$	Permission to enter into Env
	$token(T)$	Permission to progress to next time T
Query	$value(Att)$ @ T	what is the value of Att at time T
	$prog(value(Att))$ @ $Ti...Tj$	progress of $value(Att)$ during $Ti...Tj$.
Answer	$prog(value(Att, Vals))$ @ I	progress of $value(Att)$ is $Vals$ during I.
	$value(Att, V)$ @ T	value of attribute Att is V at time T
Inform	$entrance(A, Cls, Loc, IField)$	Place A of Cls at Loc with $IField$
	$entrance(confirmed)$	Confirm the entrance of the sender
	$entrance(not\_possible)$	Inform entrance was not possible
	$inf(new\_agent, Class, \Lambda_0)$	Agent is a new instance of $Class$
	$inf(\Lambda_{hist}, Cls, im\_out)$	Agent moved out of the environment.
	$no\_influence$	Agent exerts no influence on receiver.

Table 6.2: Kinds of messages a reactive *ecoagent* is allowed to send.

In what follows, PI_{ins} is the set of individuals of those classes of potential influences currently present in the environment, $I = Ti...Tj$, T is a time stamp

	MESSAGE RECEIVED	INTERPRETATION
Request	$entrance(denied, Reason)$	Entrance denied because of $Reason$
	$token(allowed, T)$	Permission to execute changes at time T
	$token(denied, out(T))$	token denied and agent is out at time T
Query	$value(Att)$ @ T	what is the value of Att at time T
	$value(Att, V)$ @ T	value of attribute Att is V at time T
	$prog(value(Att, Vals))$ @ I	progress of $value(Att)$ is $Vals$ during I.
Inform	$entrance(allowed, T, PI_{ins})$	Allowed to enter at time T within PI_{ins}
	$inf(new\_agent, Class, \Lambda_0)$	Agent is a new instance of $Class$
	$inf(\Lambda_{hist}, Cls, im\_out)$	Agent was moved out of the environment.
	$no\_influence$	Agent exerts no influence on the receiver.

Table 6.3: Messages a reactive agent is allowed to receive.

6.4.4 Messages Allowed to Active *Ecoagents*

The kinds of messages allowed to active *ecoagents* may vary depending on the application domain. In [Haddadi 95] it is suggested as an example of these message the

following ones: *demand, deny, request, inform, answer, command, accept, reject, propose, report.* I will not show the syntax of these in detail, mainly because it is not the purpose of this thesis to investigate active *ecoagents* in depth. This has been covered in other parts of the literature (see Section 5.8).

6.5 A Model for Ecoagent Based Systems

6.5.1 An Ecoagent Architecture

An *ecoagent* architecture is defined as follows.

Definition 6.2 (Ecoagent Architecture) *An ecoagent architecture is a tuple*

$\mho = \langle \mathcal{N}_{eco}, \Sigma_\Lambda, \mathcal{P}_{inf}, \mathcal{F}_{inf}, \mathcal{A}_{res}, \mathcal{I}_{act}, \Sigma_\Theta, \mathcal{S}_\Theta, \mathcal{SC}_\Theta, \mathcal{S}_\Gamma, \mathcal{M} \rangle$,

where

- $\mathcal{N}_{eco} = \mathcal{N}_{cls} \cup \mathcal{N}_{ag} \cup \mathcal{N}_{Att} \cup \mathcal{N}_{ler}$ *is a set of names, where* $\mathcal{N}_{cls}$ *is a set of class names,* $\mathcal{N}_{ag}$ *is a set of instance names (agents' identification),* $\mathcal{N}_{Att}$ *is a set of attribute names and* $\mathcal{N}_{ler}$ *is a set of names of local environmental relations,*
- Σ_Λ *is the set of all possible sets of attributes (or states),*
- $\mathcal{P}_{inf}$ *is a set of potential influences*
- $\mathcal{F}_{inf}$ *is a set of functions of influence upon agents' processes,*
- $\Sigma_{\mathcal{A}_{res}}$ *is a set of resource assimilation functions related to resource assimilation factor (RAF),*
- $\mathcal{I}_{act}$ *is a set of internal actions or computational functions associated to actions,*
- Σ_Θ *is the set of all possible local environment* Θ.
- $\mathcal{S}_\Theta$ *is a function for sensing the environment and from it generate the agent's local environment (see Section 6.8.2),*
- $\mathcal{SC}_\Theta$ *is a function for sensing changes in the agent's local environment (see Section 6.8.2),*

- $\mathcal{S}_\Gamma$ *is a function for sensing information resource (see Section 6.8.3),*
- $\mathcal{M}$ *is a* **NatureTime** *temporal model, and so actions are ordered according to their granularity of time,*

A multi-ecoagent architecture is then $\mathcal{M}_\mho = \langle \Sigma_\mho, \textcircled{\mho}, \mathcal{K}_{ext} \rangle$, where $\Sigma_\mho$ is a set of *ecoagents*, $\textcircled{\mho}$ is a distinguished special agent for representing properties of the environment (i.e. *envagent*), $\mathcal{K}_{ext} = (\Omega, \overrightarrow{out}, \overleftarrow{in})$ is the *Environmental Medium* to represent external knowledge and interaction among agents, where Ω is the *medium of communication* represented as a list of messages, and $\overrightarrow{out}/\overleftarrow{in}$ is a function which maps the set of messages an agent may send/receive and Ω to Ω. This is defined as follows.

Definition 6.3 (Communication Functions) *Let $\Sigma_{\mathcal{M}_{in}}$ and $\Sigma_{\mathcal{M}_{out}}$ be the sets of messages an agent may receive and send, respectively, E be an envagent, Ω_E be the list of messages of E, and $G \subseteq E$. Then,*

- $\overrightarrow{out}$: $\Sigma_{\mathcal{M}_{out}} \times \Omega \to \Omega$, *which is defined as follows.*

  $\overrightarrow{out}\ (send(Msg), \Omega_E) = \{Msg\} \cup \Omega_E.$

  $\overrightarrow{out}\ (broadcast(M), \Omega_E) = \Omega_E \cup \{msg(A, M) \mid A \in E\}.$

  $\overrightarrow{out}\ (multicast(G, M), \Omega_E) = \Omega_E \cup \{msg(A, M) \mid A \in G\}.$

  $\overrightarrow{out}\ (write\_ek(Desc), \Omega_E) = \Omega_E \cup \{Desc\}.$

- $\overleftarrow{in}$: $\Sigma_{\mathcal{M}_{in}} \times \Omega \to \Omega$, *is a partial function defined as*

  $$\overleftarrow{in}\ (read(Msg), \Omega_E) = \begin{cases} \Omega_E \setminus \{Msg\}, & \text{iff } Msg \in \Omega_E; \\ \Omega_E, & \text{otherwise.} \end{cases}$$

  $\overleftarrow{in}\ (wait(Msg), \Omega_E) = \Omega_E \setminus \{Msg\}$ *only when* $Msg \in \Omega_E$.

  $\overleftarrow{in}\ (wait\_rd(Msg), \Omega_E) = \Omega_E$ *only when* $Msg \in \Omega_E$.

The meaning of the communication predicates an agent uses in relation to its partial model is defined as follows.

Definition 6.4 (Interpretation of Writing Interface and Messages) *Let E be an envagent, Msg a well formed message envelope, A and B be two ecoagents of E. Then the interpretation of messages is defined as follows.*

- *write_ek(Desc) is true in A's model*

  when the execution of $\overrightarrow{out}(write\_ek(Desc), \Omega_E)$ *succeeds.*

- *send(msg(B, Msg)) is true in A's model*

  when the execution of $\overrightarrow{out}(send(Msg), \Omega_E)$ *succeeds.*

- *broadcast(Msg) is true in A's model*

  when the execution of $\overrightarrow{out}(send(msg(B, Msg)), \Omega_E)$ *succeeds for all* $B \in E$.

- *multicast(G, Msg) is true in A's model*

  when the execution of $\overrightarrow{out}(send(msg(B, Msg)), \Omega_E)$ *succeeds for all* $B \in G$.

- *read(msg(A, Msg)) is true in A's model*

  if $Msg \in \Omega_E$ *in the execution of* $\overleftarrow{in}(read(Msg), \Omega_E)$, *and*

  otherwise it is false.

- *wait(msg(A, Msg)) is true in A's model*

  only when the execution of $\overleftarrow{in}(wait(Msg), \Omega_E)$ *succeeds.*

  While this does not happen A is not allowed to execute anything at all.

- *wait_rd(msg(A, Msg)) is true in A's model*

  only when the execution of $\overleftarrow{in}(wait\_rd(Msg), \Omega_E)$ *succeeds.*

6.5.2 Informal Interpretation of Ecoagent Based Systems

An informal way to interpret *ecoagent* systems is to think in terms of partial models. This is inspired by the work [Engelfriet & Treur 94, Gavrila & Treur 94], where each agent has its own information state written according to a signature. In their work, each partial model is a mapping from the set of ground atoms to a three valued set $\{true, false, unknown\}$. Here I use *blocked* rather than *unkown*.

The basic idea we follow is that not all things which can be proved on a single agent local model, at a certain moment of time, may be proved by another one, unless they communicate exchanging such information. This means that communication primitives (see Section 6.4.1) link truth values of literals across partial models. Thus, the model

for the overall system is the union of the partial models of the agents taking the environmental medium into account. This is one way to tackle the complexity of reasoning about knowledge that traditional logics for knowledge raise [Fagin *et al.* 95].

Instead of giving a formal semantics for *ecoagent* based systems we shall see an informal way of interpreting them. This will be presented along with the definition for the structures associated with an *ecoagent* architecture, and also in the definition of a model for executing specifications of such systems.

6.6 An Execution Model for Ecoagent-Based Simulation

6.6.1 State of Computation of Reactive Agents

Now, we specify which knowledge should be taken into account by all the agent's processes when a given message is read from the environment, and how such knowledge evolves through the flow of time. This knowledge is called the state of computation of an agent.

Definition 6.5 (*Ecoagent*'s State of Computation) *Let A be an ecoagent, Λ @ T be the state of A's attributes at time T, T_f be the time-token A is waiting for, Γ_0 be the information resources to be acquired by A, N be the number of waiting messages associated to Γ, Sf be the number of messages received so far, I be the interval of sensing or observation, Θ_A be A's local environment and $\mathcal{Q}$ be the list of messages to attend from the environment. Then, the* state of computation *of A is the structure*

$$\epsilon(\Lambda \text{ @ } T, T_f, ra(N, Sf, \Gamma \text{ @ } T...T_j), \Theta_A, \mathcal{Q}).$$

Note that the history of the agent's attributes is kept as part of its partial model. One may assume that it is kept in a specific structure for this purpose, *e.g.* a database. In this structure, what could be considered as shared knowledge among all processes of an agent are:

- Θ - because whenever an OP of an agent starts to seek for resource, it needs to know which sort of resource it is going to look for within the agent's local environment.

- T_j - because at every change of state of an agent, its OPs need to seek for resource just until the closest time stamp earlier than its next time stamp.

- Σ_Γ - because whenever an OP acquires a resource the current number of acquired resources (Sf) has to be updated, and such an acquired resource has to be stored for further computation.

An agent in the real world has many sensors working in parallel, so it is reasonable to consider a computational agent as endowed with similar capabilities. The question is how an agent is going to interact with the environment by means of messages it sends and that it may receive? This issue is addressed in the next section, and also the main control of an agent which is responsible for reading such messages and passing them to the appropriate process.

6.6.2 A General Meta-Interpreter for EABS

The meta-interpreter for *ecoagent* behaviour will not be presented in the same Prolog style as the one presented in Chapter 4. A graphical representation will be used, where square boxes gather computations related to the partial model of the agent, round boxes are used to represent use of communication primitives or processes that make use of them, and an hexagon is used to represent a constraint being tested. The solver, called $active\_agent/2$, needs two names $A \in \mathcal{N}_{ag}$ and $Env \in \mathcal{N}_{cls}$. I assume that the agent is already connected to a $\mathcal{K}_{ext}$ using some computer network protocol. Its description is depicted in Figure 6.2, where

- $may\_influence(Atts, A, \Pi_A)$ associates the attributes $Atts$ of an agent A to its set of potential influences Π_A;

- $classes\_inf(\Pi_A, CInfs)$ associates Π_A to a set $CInfs$ of names of classes of influences within it;

- $location(\Lambda_0, Loc)$ associates an agent's initial state or goal to its intended location within the environment.

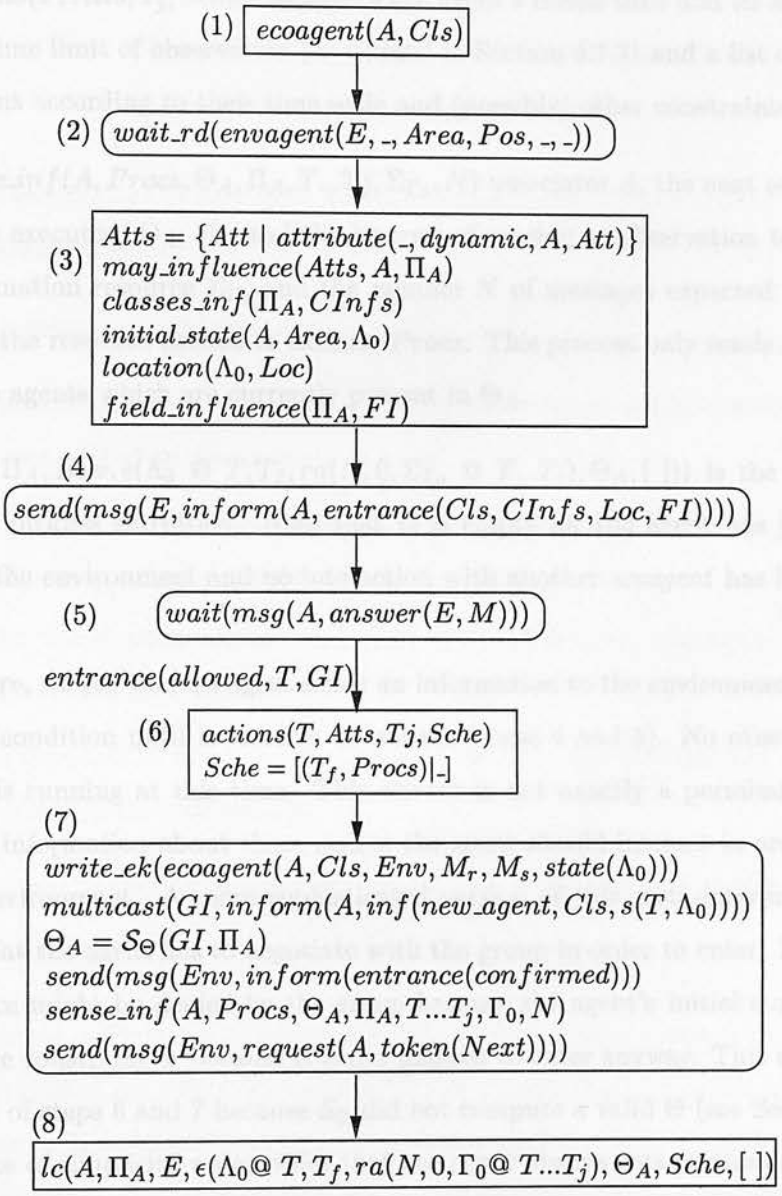

Figure 6.2: Activation meta-interpreter of an *Ecoagent*.

- $field\_influence(\Pi_A, FI)$ associates Π_A of agent A to its maximum field of influence. This is the maximum (or minimum) constraint between an attribute of A and a potential source of influence.

- $actions(T, Atts, T_j, Sche)$ associates the agent's initial time and its attributes to the time limit of observation (as defined in Section 4.7.3) and a list of scheduled actions according to their time scale and (possibly) other constraints.

- $sense\_inf(A, Procs, \Theta_A, \Pi_A, T...Tj, \Sigma_{\Gamma_0}, N)$ associates A, the next set of actions to be executed, Θ_A, Π_A and the interval of sensing or observation to the initial information resource Σ_{Γ_0} and the number N of messages expected to fulfill Γ_0 with the resource needed to execute $Procs$. This process only sends messages to those agents which are currently present in Θ_A.

- $lc(A, \Pi_A, Env, \epsilon(\Lambda_0 \ @ \ T, T_f, ra(N, 0, \Sigma_{\Gamma_0} \ @ \ T...T_j), \Theta_A, [\ ]))$ is the agent's *life cycle* process activation. Note that $\mathcal{Q}$ is empty for the agent has just entered into the environment and no interaction with another *ecoagent* has happened.

In this figure, we see that an agent sends an information to the environment and stays in waiting condition until it receives an answer (steps 4 and 5). No other process of the agent is running at this time. This answer is not exactly a permission to enter but rather information about those agents the agent should interact in order to build its local environment. A more sophisticated version of this meta-interpreter should consider that the agent has to negotiate with the group in order to enter. In this case, the entrance might be denied by the group because the agent's initial state does not satisfy some constraint or because it is not allowed to enter anyway. This would imply in the split of steps 6 and 7 because $\mathcal{S}_\Theta$ did not compute a valid Θ (see Section 6.8.2). For the sake of simplicity we consider that the agent always gets permission to enter.

In the answer from *envagent* the agent receives the current time of the environment and the group of agents belonging to its classes of influence which are currently present. In the next two steps the agent sets up its initial scheduled actions. The agent multi-casts its entrance to such a group and then $\mathcal{S}_{\Theta_A}$ is evoked. The next step is to commit its external knowledge (or interface) to the environment, and after that it sends its sensing

messages to those members of its local environment. The last step of this phase is to call the agent's life cycle process.

In this next phase an *ecoagent* reads any acceptable (or understandable) message and reacts properly. If the message is not recognizable the agent performs no action (or might reply it does not understand it) as far as this work is concerned. During this cycle the agent may either:

- read the global clock and unify it with the *time–token* it had requested to progress its internal state, or
- read a message, which can be
  - an answer for its sensing messages, then the agent assimilates the a resource represented in the contents of it, or
  - a query sent by other agents, the sent try to attend the query os stack it if the query can be answered in the future, or
  - an information about changes in the environment, the agent updates its local environment, or
  - a negation to progress its behaviour, then the agent attend any message left and stop its computation.
- attend unattended messages if the global time is not equal to its requested *time–token*.

This is depicted in Figure 6.3, where $startlife\_cycle$ is a shor notation for the expression $lc(A, \Pi_A, Env, \epsilon(\Lambda_0 \ @ \ T, T_f, ra(N, 0, \Sigma_{\Gamma_0} \ @ \ T...T_j), \Theta_A, [\,]))$, $attend(M, \epsilon, \epsilon')$ executes the attendance of Q or store it in $\mathcal{Q}$ of state ϵ updating it to ϵ', $change\_state(\epsilon, \epsilon')$ represents the execution of computations or message exchange needed to change the state of the agent (this process is explained in Figure 6.4), $update(\epsilon, \Sigma_{\Gamma'}, \epsilon')$ updates the state of computation given the new information resource, $final\_attendance(T, \epsilon)$ reads all queries, requests or informs for the agent in Ω_E and makes the final attendance of messages in ϵ. Then the agent informs *Env*, which should be waiting for its message, that it is out.

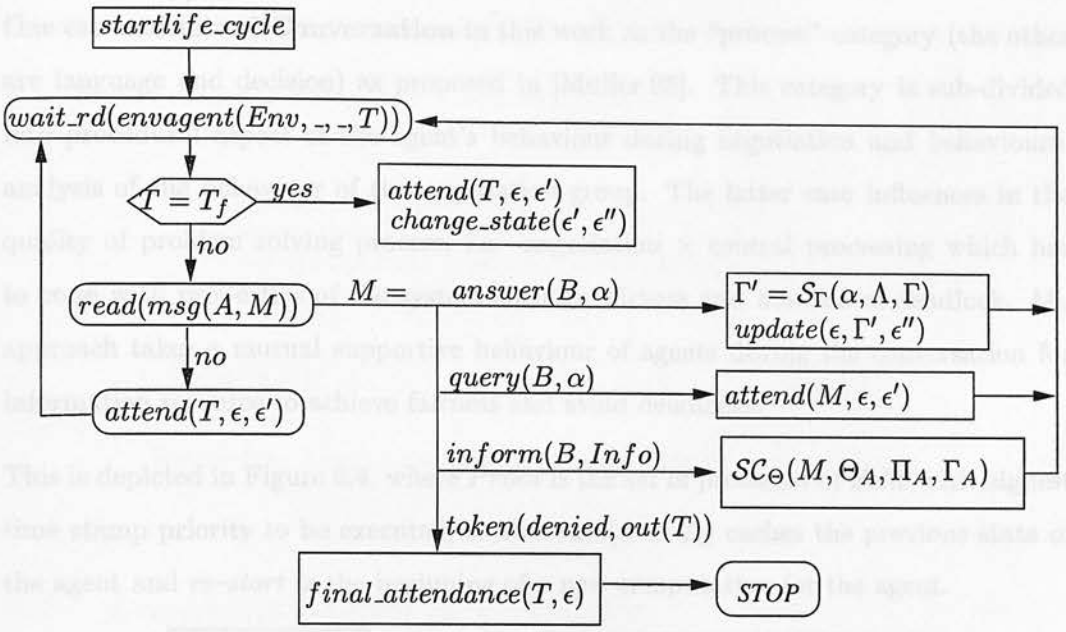

Figure 6.3: Reactive agent's Life Cycle or Main Activity.

Although an agent stops its activity when receiving an "order" to get out of the environment, it could simply leave the environment and look for another one. This will depend on the type of reactive agent we are modelling. For instance, a tree can not leave its environment except if it is cut off, making it stop its activity, but an animal could simply leave the environment if it receives a message saying some malentity is putting in danger its life. The agent may stop its activity but not before attending to any remaining messages. This mechanism prevents starvation followed by deadlock, in the case another agent recently entered into the environment is waiting for an acknowledgement from the agent.

Once permission to progress is granted the agent first attends any query or request left un-attended. Then the agent may progress to its next state or stays in a state called **Conversation**, where it is going to either sense external information, or attend a new query or update its local environment with the news that some agent left and sent a last information to it. The **Conversation** proposed here is not meant to solve conflicts or to build a plan for cooperative action as some systems do, *e.g.* [Kreifelts & vonMartial 91, Rosenschein & Zlotkin 94]. In this case the conversation is usually called negotiation.

One can understand **Conversation** in this work as the "process" category (the other are language and decision) as proposed in [Muller 96]. This category is sub-divided into procedural aspect of the agent's behaviour during negotiation and behavioural analysis of the behaviour of the negotiation group. The latter case influences in the quality of problem solving process, i.e. negotiation × central processing which has to cope with properties of the system such as fairness and absence of deadlock. My approach takes a mutual supportive behaviour of agents during the conversation for information resource to achieve fairness and avoid deadlocks.

This is depicted in Figure 6.4, where *Procs* is the set of processes of *Sche* with highest time stamp priority to be executed, $cache\_state(\Lambda\ @\ T_i)$ caches the previous state of the agent and *re-start* is the beginning of a new computation for the agent.

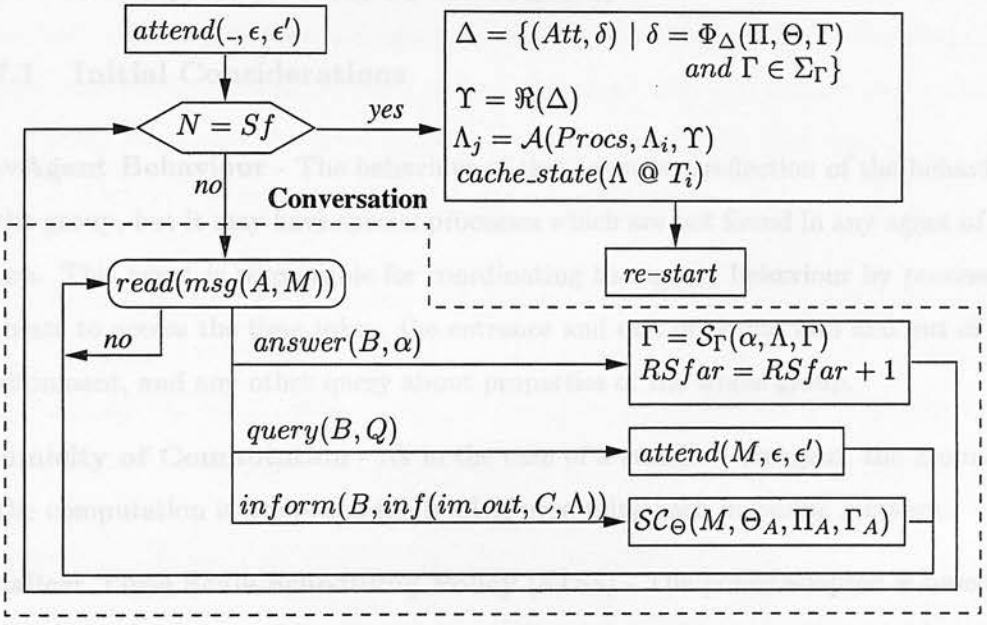

Figure 6.4: Agent's change of state and **Conversation** phase.

The *re-start* process consists basically of planning a new schedule of actions if the agent "decides" to keep running in the current *envagent*. In this case, it is going to sense its local environment and ask a new *time-token*, and then enters into its life cycle once again. Note that the decision an agent makes about staying or leaving the environment will depend on its capabilities to do so, i.e. depend on its level of autonomy. Figure 6.5 shows this phase for a reactive *ecoagent* where it is assumed there is no autonomy, i.e.

the freedom to leave the environment which would not be the case for active agents.

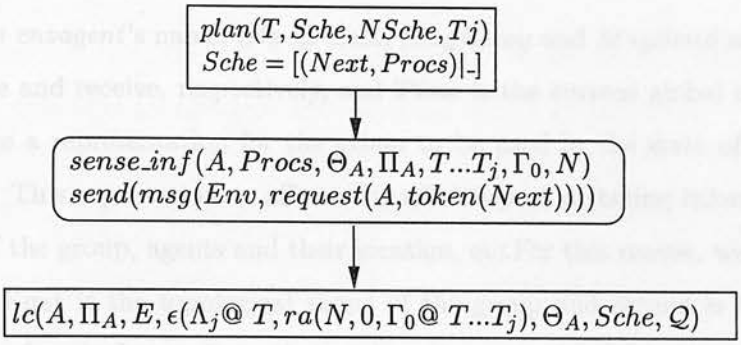

Figure 6.5: Agent's re-starting process after changing its state.

6.7 *Envagent* Model of Execution

6.7.1 Initial Considerations

EnvAgent Behaviour - The behaviour of this agent is a reflection of the behaviour of the group, but it may have special processes which are not found in any agent of the group. This agent is responsible for coordinating the agents behaviour by processing requests to access the time token, the entrance and exit of agents into and out of the environment, and any other query about properties of the whole group.

Atomicity of Computation - As in the case of a standard ecoagent, the atomicity of the computation is limited to the level of processing each incoming message.

Smallest Time Scale Scheduling Policy (STSS) - The policy adopted is based on the time scales of the agents. But *envagent* does not need to know the actual scale of its components. Each one of them just sends the time stamp of the top processes in its scheduler of actions. The *envagent* organises the requests according to the smallest time stamp.

6.7.2 Group Representation

The internal representation of a group corresponds to the declarations as presented in Table 6.1 for *ecoagent*. The external knowledge (or Interface) for an *envagent* will be

$envagent(Env, Cls, MsgRecog, MsgSend, Time)$

where Env is *envagent*'s name, Cls its class, $MsgRecog$ and $MsgSend$ are messages it can recognise and receive, respectively, and $Time$ is the current global time. We now have to chose a representation for the group to be used in the state of computation of an agent. This representation affects the efficiency of accessing information about properties of the group, agents and their location, etc.For this reason, we will consider two things. First is the topological shape of the group and second is the structural organisation of such shape. Note that such notions are not related to geometrical or geographical shape and positions, but in more general terms. For example, the location of an agent within the structure of a company.

I consider that the environment is sub-divided into zones and that each zone is uniquely identified by its boundaries and agents (grouped in classes) within it. The shape and boundaries are dependent on the class of *envagent*. For our purpose we assume a rectangular shape and the boundaries are its left-bottom (LB) and right-upper (RU) corners. This will be represented by the following structure.

Definition 6.6 (Group Structure of Ecoagents) *Let* $Ags_1, \ldots, Ags_j, \ldots, Ags_k$ *be sets of ecoagent names,* $Cls_1, \ldots, Cls_j, \ldots, Cls_k$ *be the classes of* $Ags_1, \ldots, Ags_k$*, respectively,* $LB_1, \ldots, LB_i, RU_1, \ldots, RU_i$ *be points in the environment. A* Group Structure *is the structure*

$$\mathcal{G} = group(N, [(zone(LB_1, RU_1), [(Cls_1, Ags_1), \ldots, (Cls_k, Ags_k)]),$$
$$\ldots$$
$$(zone(LB_i, RU_i), [(Cls_j, Ags_j), \ldots, (Cls_k, Ags_k)])])$$

where N *is the total number of agents over each* $Ags_i, i = 1, \ldots, k$.

Note that the second argument of the term $group/2$ is not necessarily a list data structure. The zones' boundaries can be used as key for a fast access, for example.

6.7.3 Envagent's Mechanism of Communication and Messages

The mechanisms of communication for an envagent are the same as those of an agent. The kinds of messages an envagent may send are shown in Table 6.4.

	MESSAGE TO SEND	PURPOSE
Answer	$entrance(allowed, T_c, IndInf)$	Grant entrance into Env
	$token(allowed, T)$	Receiver get time–token at T
	$token(denied, move\_out)$	Receiver must get out
	$value(Att, V)$ @ T	Value of group attribute at T
Demand	$offer(Amount, Res)$ @ T	offer $Amount$ of Res

Table 6.4: Messages an *envagent* may send.

The possible kinds of requests and queries an envagent may receive, as far as the domain of ecological modelling is concerned, are shown in Table 6.5.

	MESSAGE TO RECEIVE	PURPOSE
Inform	$entrance(A, Cls, Loc, IField)$	Place A of Cls at Loc and $IField$
	$move\_to(Cls, CInfs, Pos, IField)$	Place A of Cls at new location
	$confirm\_move$	Agent has effectuated movement
	not_moved	Agent remains where it is
Request	$token(T)$	Permission to progress to time T
Query	$value(Att)$ @ T	Value of group attribute at T
Command	$out(ecoagent(Agent, Cls))$	Event to move $Agent$ out of Cls
	$halt$	Halts the whole system

Table 6.5: Messages an *envagent* may send.

Usually, a query is temporal and if the time for which it should be answered has not come, then *envagent* put it in its list of unattended requests. In the case of non time dependent queries, they are assumed to be related to the current state of the world, and so may not need to be stacked by the *envagent*.

6.7.4 Scheduler of a Group of Ecoagents

In Section 5.6.3 we saw the structural similarities between the scheduler of an individual and a group of agents. The representation of such a scheduler is not a direct translation as it can be at the individual level. The reason is because we want to take advantage of the number of agents within the environment and the number of requests associated to

each time token requested. This will help us to use a policy which is free of starvation and deadlock (see Section 6.7.6). The scheduler is defined as follows.

Definition 6.7 (Envagent Time Stamp Scheduler of Processes) *Let $T_1, \ldots, T_n$ be an ordered sequence of time tokens requested for the sets of agents $Ags_1, \ldots, Ags_n$, $m_1, \ldots, m_n$ be the number of elements of sets $Ags_1, \ldots, Ags_n$, respectively. Then, the* Envagent Time Stamp Scheduler of Processes *is the structure*

$$\wp = [(T_1, m_1, Ags_1), \ldots, (T_n, m_n, Ags_n)]$$

6.7.5 Envagent State of Computation

The envagent state computation will carry information about the current time, the total number of agents, a tuple containing the number of requests and a list of requests and their respective time stamps. Time stamps associated with messages, in general, will always be a moment of time, i.e. a time stamp is not allowed to be an interval or a collection interval. More formally we have.

Definition 6.8 (Envagent State of Computation) *Let T be the current global moment of time, $\mathcal{G}$ be a group structure of an envagent E, $\wp$ be a list of time token requests and N_r the total number of requests in it, $\mathcal{Q}$ be a list of queries about properties of the whole environment, E_{act} be a set of external events (or actions) which may change the constitution or structure of E. Then, the* Envagent State of Computation *is the structure* $\Im = s(T, \mathcal{G}, req(N_r, \wp), \mathcal{Q}, E_{act})$

Whenever an *envagent* receives a query about some property of the environment it has to keep information about this query after broadcasting the query to its components. Such information is kept in $\mathcal{Q}$ and has the following form, where A is the agent which sent the query, $GrAttr$ is the group attribute (or property) at time T, N is the number of components to which *envagent* has to wait for an answer about such a property, NSf is the number of agent which have answered so far, and Γ is the resource information of *envagent* associated with $GrAttr$.

$$(A, value(GrAttr) \ @ \ T, N, NSf, \Gamma)$$

6.7.6 Time Token Distribution Policy (TTDP)

We already know that an envagent uses a STSS policy for scheduling the requests according to their time scale. Actually the time scale is used by each agent to compute their time stamp at which the request should be attended. But as there is no concept of real clock, the envagent does not simulate any clock. So, when should the time token be liberated? Suppose the state of computation is

$$s(Time, group(N, Agents), req(NReq, [(T_1, N_1, Reqs_1), \ldots, (T_n, N_n, Reqs_2)]), \mathcal{Q}, E_{act}),$$

where $NReq = N_1 + \ldots + N_n$, and $NReq \leq N$. Then, the requests should be answered, or resources liberated whenever $NReq = N$. The liberation of *time-token*, however, is not going to be done by explicit message passing to all agents on the top of the scheduler (i.e. those in $Reqs_1$). Those agents just need to read the external interface of *envagent*, and this can be done concurrently (or in parallel) by the sensors of the agents. Thus, such a liberation means that *envagent* updates the value of the global clock.

Other agents in the rest of the scheduler will have to wait until they read a global clock which unifies with theirs. If they have to wait for this, it is because they are supposed to work at coarser level of time scale than those in $Reqs_1$ or are not aligned through the flow of time. When those in $Reqs_1$ have processed their computation they will either ask for the token again or be moved out of the group. In the first case, their requests will be inserted according to their time stamp, i.e. we make insertion by order. In the latter case, as the total number of requests $NReq$ will decrease, those waiting will be answered because $NReq = N$ may hold, depending on the number of agents which are moved out of the group.

One important thing we have to bear in mind for the specification of processes associated with those messages an *envagent* may receive, is that the behaviour of agents which are members of a group is not fully dependent of the answer the envagent will give to them. By this we mean that while an agent asks the token to progress its state from one time to another, it stays active seeking for information or resource it may needs, such as other agents' attributes.

6.7.7 *Envagent* General Meta-Interpreter

A general meta-interpreter for an *envagent* is shown in Figure 6.6. The predicate $active\_envagent(E)$ activates *envagent* E (assuming it is already connected to a network as a client. The $\mathcal{I}_{cond}/2$ sets the initial conditions or state of the *envagent* according to the specification given by the modeller (geographical position, area, etc.). The $starting\_time/1$ gets the initial time of the model of time specified, and after writing its external interface into $\mathcal{K}_{ext}$ the agent enters in its life cycle where it will be waiting for messages to processing them.

a)	*b)*
$active\_envagent(E) \Longleftarrow$ $envagent(E, Class)$ $\mathcal{I}_{cond}(E, C_{env})$ $starting\_time(T)$ $write\_ek(envagent(E, Class, C_{env}, T)$ $env\_lc(E, s(T, group(0, [\,]), req(0, [\,]), [\,], [\,]))$.	$env\_lc(E, \Im) \Longleftarrow$ $wait(msg(E, Msg))$ $(\neg Msg = command(\_, halt)$ & $process(Msg, E, \Im, \Im')$ & $env\_lc(E, \Im')$ $\vee$ $Msg = command(\_, halt)$ & $broadcast(move\_out)$).

Figure 6.6: *Envagent* general meta-interpreter.*a*) activation steps which ends with call to *b*) *envagent* life cycle.

An *envagent* may be specified to recognise and process a great number of messages. In what follows I show a possible behaviour such an agent may have according to a very limited set of messages as defined above (this limitation is not a condition, but rather an example). In what follows, $zones\_influenced(Loc, RI, Zones)$ associates an agent's location and its ratio of influence to the zones within *envagent*'s topology which have intersection with its ratio; $insert(A, B, C)$ insert A into set B resulting in C; $agents\_inf(Zones, \mathcal{G}, CInfs, AInf)$ associates a set of zones in the *envagent* area, the whole group $\mathcal{G}$ and a set $CInfs$ of classes of influence to a set of agents $AInf$ of $CInfs$ which are currently present in the $Zones$; and $insert\_into\_group(A, Loc, Cls, \mathcal{G}, \mathcal{G}')$ associates A, A's location Loc, A's class Cls and a group $\mathcal{G}$ to a new group $\mathcal{G}'$ added with A.

$process(inform(A, entrance(Cls, CI, Loc, RI)), E, s(T, \mathcal{G}, Reqs, \mathcal{Q}, E_{act}), \Im') \Longleftarrow$

$zones\_influenced(Loc, RI, Zones)$ &
$insert(Cls, CI, CInfs)$ &
$agents\_inf(Zones, \mathcal{G}, CInfs, AInf)$ &
$send(msg(A, answer(E, entrance(allowed, T, AInf))))$ &
$wait(msg(E, inform(A, Info)))$ &
(($Info = entrance(not\_possible)$ & $\Im' = s(T, \mathcal{G}, Reqs, \mathcal{Q}, E_{act})$)
$\vee$
($Info = entrance(confirmed)$ &
$insert\_into\_group(A, Loc, Cls, \mathcal{G}, \mathcal{G}')$ &
$\Im = s(T, \mathcal{G}', Reqs, \mathcal{Q}, E_{act})$)).

In *apply_policy*/3 below, if the condition for time token distribution is satisfied (i.e. $NReq = N$). Then the *envagent* will first look at the E_{act} to see if there is any external event which may cause the progress of some agent to be stopped or cause any sudden change on its state. Only after doing this the token is distributed, if it is the case.

$process(request(A, token(T)), E, s(Tc, \mathcal{G}, ReqSf, \mathcal{Q}, E_{act}), \Im') \Longleftarrow$
$insert\_req(E, A, T, ReqSf, NReqSf)$ &
$apply\_policy(E, s(Tc, \mathcal{G}, NReqSf, \mathcal{Q}, E_{act}), \Im')$.
$process(command(B, out(ecoagent(A, Cls))), E,$
$s(T, group(NE, Ags), req(N_r, \wp), \mathcal{Q}, E_{act}), \Im') \Longleftarrow$
$get\_agents(Cls, Ags, AgentsCls)$ &
($A \notin AgentsCls$ &
$send(msg(B, answer(E, no\_agent(A, Cls))))$ &
$\Im' = s(T, group(NE, Ags), req(N_r, \wp), \mathcal{Q}, E_{act})$
$\vee$
$A \in AgentsCls$ &
$\neg already\_moved(A, Cls, E)$ &
$\Im' = s(T, group(NE, Ags), req(N_r, \wp), \mathcal{Q}, [out(ecoagent(A, Cls))|E_{act}])$).
$process(command(\_, out(ecoagent(\_, \_))), \_, \Im, \Im)$.

The remaining specification is related to the queries an *envagent* receives about properties of the environment. The query is multi-casted to the group and the agent keeps

information about it on its list of queries as said above. These are specified as follows, where $|G|$ is the number of elements of set G.

$process(query(A, value(Att)\ @\ T), E,$
$\quad s(T_c, group(N, Agents), Req, [\,], E_{act}),$
$\quad s(T_c, group(N, Agents), Req, [(A, value(Att)\ @\ T, N, 0, \Gamma_0)], E_{act})) \Longleftarrow$
$\quad may\_influence(\{Att\}, E, \Pi_E)\ \&$
$\quad Inds = \{A \mid A \text{ is of class } Cls \text{ and } (value(Att, agent(Cls)), \_, \_) \in \Pi_E\}\ \&$
$\quad multicast(query(E, inf(value(Att)\ @\ T, \_, \_)), Inds)\ \&$
$\quad N \text{ is } |Inds|.$

Finally, we just need to specify the way in which the group attribute is computed. The functions used are similar to the ones used for an *ecoagent*. The only difference is that *envagents* do not have local environmental relations for the sake of simplicity. A more elaborate architecture to deal with dynamics among groups of agents could consider this. Then, our final specification is as follows, where *dispatch*/4 process an answer and test whether or not the group attribute factor has been completed.

$process(answer(A, inf(value(Att, V)\ @\ T, \_, \_)), E,$
$\quad s(Tc, Group, Req, \mathcal{Q}, E_{act}), \Im') \Longleftarrow$
$\quad \mathcal{Q}' = \mathcal{Q} \setminus \{(B, value(Att)\ @\ T, N, Sf, \Gamma)\}\ \&$
$\quad \Gamma' = \mathcal{S}_\Gamma(answer(A, inf(value(Att, V)\ @\ T, \_, \_)), \_, \Gamma)$
$\quad NSf \text{ is } Sf + 1\ \&$
$\quad dispatch(E, (B, value(Att)\ @\ T, N, NSf, \Gamma'), \mathcal{Q}', \mathcal{Q}'')\ \&$
$\quad \Im' = s(Tc, Group, Req, \mathcal{Q}'', E_{act}).$
$dispatch(E, (B, value(Att)\ @\ T, N, N, \Gamma), \mathcal{Q}, \mathcal{Q}) \Longleftarrow$
$\quad may\_influence(Att, E, \Pi_E)\ \&$
$\quad \Delta = \{(Att, \delta) \mid \delta = \Phi_\Delta(\Pi_E, \_, \Gamma)\}\ \&$
$\quad \Upsilon = \Re(\Delta)\ \&$
$\quad change(P, Att)\ \&$
$\quad compute(P, F)\ \&$
$\quad (Att, Inf) \in \Upsilon \text{ holds and } execute(F, Att, Inf, \_, V)\ \&$
$\quad send(msg(B, answer(E, value(Att, V)\ @\ T))).$

$dispatch(E, (B, value(Att) \ @ \ T, N, Sf, \Gamma), \mathcal{Q}, \mathcal{Q} \cup \{(B, value(Att) \ @ \ T, N, Sf, \Gamma)\}) \Longleftarrow$
$Sf < N.$

The rest of this chapter is dedicated to the specification of a possible way to implement the dynamic knowledge and functions of an *ecoagent*, as well as to prove some important properties of the system.

6.8 Dynamic Knowledge of *Ecoagents*

6.8.1 Potential Influences: Dynamic Structure

In Section 5.3.1 was proposed a language for defining the potential sources of influences upon an agent's behaviour. It was said that it might be useful to represent that knowledge in a dynamic structure, so that the architecture could be easily extended to allow changes over long periods of time. In what follows Σ_v, Σ_h and Σ_{inf} are the sets of terms according to **Definition 5.1** of Section 5.3.1.

Definition 6.9 (Dynamic Structure for $\mathcal{P}_{inf}$) *Let A be an agent, Att be an attribute of A, $Value_i \in \Sigma_v, Holds_i \in \Sigma_h, Func_{inf_i} \in \Sigma_{inf}$ and P_i is a process of A $(i = 1, \ldots, n)$. A* Dynamic Structure *for $\mathcal{P}_{inf}$ over attribute Att is a structure*

$$\Pi = [(Att, [(Value_1, Holds_1, Func_{inf_1}), \ldots (Value_n, Holds_n, Func_{inf_n})])]$$

such that $change(P_i, Att) \wedge affect(P_i, A, Value_i, Holds_i, Func_{inf_i})$ is true, for $i = 1, \ldots, n$. Σ_Π is the set of elements of this form.

Π is generated from the sensing ability of an agent, given it local environment and resource information. Now, we can give a more detailed definition of sensing than the general one proposed in Section 5.3.3. The main difference is that the external information Ψ will be represented by the contents of the message read by the agent.

6.8.2 Local Environment: Data Structure

The local environment (LE) is a dynamic structure which is sensible to the entrance, the departure of an *ecoagent* into and from the environment as well as to changes their

components may experience. There are three things of extreme importance for an agent's LE. First, is the identification of the members of LE. Second, is the nature of the relation between every member and the agent (see Section 5.3.2 for some examples). Third, the information associated with the value of the relation between the agent and a member of its LE.

An agent will always assume that those members of its local environment are present unless they inform they are going out or moving for some reason. This assumption is confirmed whenever the agent receive an answer to its sensing messages. A counterpart of it is that every agent must inform those members of its local environment whenever they have to leave it the . This policy guarantees that the agent will dispatch messages only to those which are currently present. Also, this will avoid problems of deadlock during the computation of state transition of the environment (see Section 6.8.5).

Definition 6.10 (Local Environment) *Let A be agent, $A_1, \ldots, A_j, \ldots, B_k, \ldots, B_n$ are agents which exerts some influence on A's behaviour, $\rho_1, \ldots, \rho_n$ be local environmental relations and $Inf_1, \ldots, Inf_j$ external information about $A_1, \ldots, A_j$, and $Inf_k, \ldots, Inf_n$ external information about $B_k, \ldots, B_n$. Then, the* Local environment *of A is a structure*

$$\Theta = [(\rho_1, [(A_1, Inf_1), \ldots, (A_j, Inf_j)]),$$
$$\ldots$$
$$(\rho_n, [(B_k, Inf_k), \ldots, (B_n, Inf_n)]),$$

Since Θ is dynamically updated it depends on the interactions between the agent and its local and whole environment. The first instance of it appears when the agent enters into the environment and communicates with those agents which are candidates to influence it. The set of such candidates are given by *envagent* which keeps a list of classes of influence (see 5.6.2). In the Algorithm 4, $eval(C, V_1, V_2)$ is true if the constraint C holds between V_1 and V_2.

When the agent is already part of a running environment the second case that Θ may change is then related to transformations in the topology and/or organisation of the environment. The agent may then sense either the entrance of a new component or the

Algorithm 4 Sensing Environment

Require: A is an ecoagent of class Cls, Λ_0 is the initial state of A, G is a set of ecoagents of class of influence $CInfs$ to which A has multi-casted the message $inform(A, inf(new\_agent, Cls, \Lambda_o))$, Σ_Π is a set of potential influences, Σ_Θ is the the set of local environments, and $\mho$ the partial model of A.

Base case: $\mathcal{S}_\Theta([\,], \_, \Theta_A) = \Theta_A$

Recursion: $\mathcal{S}_\Theta([B|R], \Pi_A, \Theta_A) = \Theta'_A$ if

$wait(msg(A, inform(B, Inf)))$

if $Inf = no\_influence$ **then**

$\Theta'_A = \mathcal{S}_\Theta(R, \Pi_A, \Theta_A)$

end if

if $Inf = rel(CInfs, Rel, V_r)$ and $(affect(\_, A, value(Att, agent(CInf)),$
$holds(Op, value(Rel, \_, \_), V) \rightarrow \rho, \_)) \in \Pi_A$ and
$eval(Op, V_r, V)$ holds **then**

$(\rho, Le) \in \Theta_A$

$\Theta'_A = \Theta_A \setminus \{(\rho, Le)\} \cup \{(\rho, [(B, in, V_r)|Le])\} \cup \mathcal{S}_\Theta(R, \Pi_A, \Theta_A)$

else

$\Theta'_A = \mathcal{S}_\Theta(R, \Pi_A, \Theta_A)$

end if

departure of an agent from its local environment or the movement of an agent already part of it. For simplicity sake this work considers only the first two situations.

In the first, only the local environment may be updated and, if it is the case, the agent send a sensing message. In the second, local environment and resource information are updated. In Algorithm 5 $conditions(Cls, \Pi, Conds)$ associates a class Cls and a set of potential influences Π to the set, $Conds$, of conditions under which instances of Cls exert influence, $value(Att, \Lambda, V)$ associates Att and state Λ to the value V of Att, $eval(value(Rel, V_i, V_j), V)$ is true if *the value of the constraint relation* Rel *between* V_i *and* V_j *is* V.

Note that there may exist many other kinds of information an agent is able to understand, but these are the only ones which cause change in the local environment. Furthermore, it is not our purpose to give a detailed definition of all possible kinds of information an agent is able to handle.

Let $\mathcal{N}_{ler}$ be the set of local environmental relations, Σ_Θ be the set of local environments, Σ_{par} be an ordered set representing a domain function. The *sensing local environment* of an agent is $\mathcal{S}_\Theta : \mathcal{N}_{ler} \times \Sigma_\Theta \rightarrow 2^{\Sigma_{par}}$. Henceforth, $2^{\Sigma_{par}}$ is the set of all possible domains of the functions of influence.

Algorithm 5 Sensing Changes in the Environment

Require: A is an ecoagent of class C_1 with state Λ_A, potential influences Π_A, local environment Θ_A, information resource Σ_Γ to be acquired from T_i until T_o, B is an ecoagent of class C_2 with recent history $Hist_B$, $\mathcal{S}_\Gamma$[1] is function of the agent to sensing the external local environment.

Case 1: $\mathcal{SC}_\Theta(inform(B, inf(new\_agent, C_2, s(T, \Lambda_0))), \Theta_A, \Pi_A, \Sigma_\Gamma) = (\Theta'_A, \Sigma_\Gamma)$ if

for all $(holds(Op, value(Rel, value(Rel, value(Att_i, A),$
$value(Att_j, agent(C_2))), Cons) \rightarrow \rho) \in Cs$ **do**

$value(Att_i, \Lambda_A, V_i)$ and $value(Att_j, \Lambda_0, V_j)$

$eval(value(Rel, V_i, V_j), V)$

if $eval(Op, V, Const)$ holds **then**

$(\rho, Le) \in \Theta_A$

$\Theta'_A = \Theta_A \setminus \{(\rho, Le)\} \cup \{(\rho, [(B, V)|Le])\}$

$send(msg(B, inform(A, rel(C_1, Rel, V))))$

else

$\Theta'_A = \mathcal{S}_\Theta(R, \Pi_A, \Theta_A)$

$send(msg(B, inform(A, no\_influence)))$

end if

end for

for all $(holds(loc\_env\_rel(\rho, agent(C_2), A))) \in Cs$ $(\rho, Le) \in \Theta_A$ and $(B, \_) \in Le$ **do**

$send(msg(B, query(A, inf(progress(value(Att)) @ T...T_o, \_, \_))))$ or

$send(msg(B, demand(A, inf(amount(X, Att) @ T...T_o, \_, \_))))$

end for

Case 2: $\mathcal{SC}_\Theta(inform(B, inf(im\_out, C_2, Hist_B), \Theta_A, \Pi_A, \Sigma_\Gamma)) = (\Theta'_A, \Sigma_{\Gamma'})$ if

if $Vals_1, \ldots, Vals_n$ are the values of attributes $Att_1, \ldots, Att_n$ from T_i until T_o in $Hist_B$ **then**

$\Sigma_{\Gamma'} = \mathcal{S}_\Gamma((Att_1, C_2, Vals_1), \Lambda_A, \ldots, \mathcal{S}_\Gamma(Att_n, C_2, Vals_n))$

$\Theta'_A = \Theta_A \setminus \{(\rho, (B, \_)) \mid (\rho, (B, \_)) \in \Theta_A\}$

end if

6.8.3 Information Resource Structure

In Section 5.3.3 we saw that an agent maps external information to an internal representation called information resource. According to assumption made in Section 6.2.1, the view an agent has of Ψ (the external information) can be simply represented by the pair of messages used to request/query/etc and receive answer information about it. The following definitions capture this concept, where *information value* means a value (numerical or not) or a tuple composed of numerical, qualitative or other tuples as values (i.e. first-order predicate calculus function term or reified predicates if necessary).

Definition 6.11 (Information Resource) *Let A be an agent, $B_1, \ldots, B_n$ be ecoa-*

gents of class Cls, and $V_1, \ldots, V_n$ *be a set of information values associated with an attribute Att of each* B_i *which are of interest for A. Then the* information resource acquired *by A in relation to the influence of the attribute Att of each agent* $B_1, \ldots, B_n$ *of class Cls is the structure* $\gamma = (Att, Cls, [(B_1, V_1), \ldots, (B_n, V_n)])$. *If A has n attributes then* $\Gamma = \{\gamma_1, \ldots, \gamma_n\}$ *is the set of* information resource *information of A, where each* Γ_i *is the information resource associated with attribute* Att_i.

In order to extract information from Γ we have the following operation. Let $\mathcal{N}_{cls}$ be the set of classes of ecoagents, $\mathcal{N}_{Att}$ be the set of attribute names associated to $\mathcal{N}_{cls}$, Σ_Γ be the set of all resource information and $2^{\Sigma_{par}}$ be the set of all possible domains of the functions of influence of A. Then the *accessing information resource* is $\Phi_\gamma : \mathcal{N}_{cls} \times \mathcal{N}_{Att} \times \Sigma_\Gamma \to 2^{\Sigma_{par}}$.

Sensing is a composition of acquiring external information and give to it an internal interpretation. However one cannot generalise the interpretation function for it depends on the domain of application, the class of the agent and the sensed information. What an agent senses is the value of some attribute, either observed or absorbed, during a certain period of time. In the first case there may be more than one value, while in the second just one. In this way we need a way to say how the value is going to be interpreted and a meta-interpreter for the algebra used to implement the way in which the agent understand the external information. The only thing we may generalise here is the protocol for defining these things or a *scheme*.

Definition 6.12 (Interpretation Scheme of Ψ) *To say that the value of an attribute Att of an agent of a class C is interpreted in some way How, depending on the state* Λ *of the agent we write* $sense\_ext\_knowledge(value(Att), agent(C), \Lambda, How)$. *If* $Vals$ *is a (possibly empty) set of values observed or absorbed of Att, How is the way in which the agent interpret Att, and* $IVals$ *is a valid set of interpreted values then we write* $interpret(Vals, How, IVals)$.

The parameter *How* is the name of a computation used to map external information to the form in which the agent understands it. There can be many ways of interpreting such information, *e.g. each_one* - the agent assumes the information as it is, *average* -

the agent uses some algebra to compute the average or other computation defined by the modeller. In Algorithm 6 we have the general mechanism for sensing resource.

Algorithm 6 Sensing Information Resource: $\mathcal{S}_\Gamma : \Sigma_\alpha \times \Sigma_\Lambda \times \Sigma_\Gamma \rightarrow \Sigma_\Gamma$

Require: Σ_Λ is the set of states of an agent A, Σ_Γ is the set of all resource information for A, Σ_α is the set of $(Att, C, B, Vals)$ where $Vals$ is the set of values of attribute Att of agent B of class C which A received as an answer α to its sensing messages, and $\gamma_i = (Att, Cls, IVals) \in \Gamma_A$ associated associated to A's ith attribute.

Case 1: $\mathcal{S}_\Gamma((Att, C, Vals), \Lambda, \Gamma_A) = \Gamma_A \setminus \{\gamma_i\} \cup \{(Att, Cls, IVals) \cup \{(B, Infs)\})\}$ if $sense\_ext\_knowledge(value(Att), agent(C), \Lambda, How)$ $interpret(Vals, How, Infs)$ holds for A.

In fact, $sense\_ext\_knowledge/4$ can be an assertion or a rule. In the first case it is assumed that the agent will always sense the external information in the same way, no matter what its state Λ. In the second, the state of the agent is relevant. For example, an agent with dynamic internal attributes such as intentions and beliefs has different ways of sensing the same information.

Depending on the architecture chosen, Σ_Γ may require some period of time in order to be fully completed. This period will be longer if sequential computation is assumed within each agent, and it will be faster if for each sensor there is a processor working in parallel with others. In any case the final resource information acquisition will be then a composition of $\mathcal{S}$ (proof is given in Section 6.8.5), no matter the order in which answer to sensing messages arrive.

6.8.4 Assimilating Resource: Structures and Operations

We saw in Section 5.3.3 that *ecoagents* have a sort of assimilation function. This function depends on the state of the agent and how it will compose the set of influences received over a certain period of time. Such a function, however, is domain dependent and there is no framework which provides a general assimilation function. The most we can do is to generalise some parts of it which can be used by the architecture, but which will depend on the modeller's definition. The items which can be generalised are just interfaces (or schema of definition) for

1. the specification of functions for information resource assimilation,
2. the execution (or solver) of the algebra associated to the language used to specify

these functions,

3. the composition of many instances of the same and different functions of influence.

Definition 6.13 (Function Definition Scheme) *Let $\mathcal{N}_F$ be a set of function names, $\pi \in \mathcal{N}_F$, $\{p_1, \ldots, p_n\} \in 2^{\Sigma_{par}}$ be a sequence of parameters to which π is applied, Expr be an expression involving $p_1, \ldots, p_n$ defined in some algebra. Then a* Function Definition Scheme (FDS) *is a declaration of the form $function(\pi, par(p_1, \ldots, p_n), Expr)$, and $\mathcal{F}_{inf}$ is the signature of expressions of this form.*

This notation was used for two reasons. The first is to take advantage of unification when generating instances of such a definition. The second is because the subject of the influence in the definition of potential influences (see Section 5.3.1) is related to its position in such scheme, and so the generation of function instances is straightforward. The execution (or evaluation) interface of an expression can be similar to the usual way of defining meta-interpreters (see Section 4.6 for an example). We shall use the predicate $eval(Expr, V)$ to mean *the evaluation of Expr is* V, and if V_1 and V_2 are two resources associated with the same function of influence then $V_1 \uplus V_2$ is the composition of V_1 and V_2 specified in the algebra which executes $eval/2$.

To make use of the function definition scheme it is necessary to supply intermediate structures to allow its straight use by unification. Such structures should represent information extracted from the set of potential influences (position of parameters) and information resource from Θ and Γ. A formal description of them is given as follows, where p_{ij}^k stands for the *i–th* parameter of the *j–th* instance of the *k–th* function.

Definition 6.14 (Influence Function Parameter Schema) *Let $\pi_1, \ldots, \pi_n \in \mathcal{N}_F$ be functions of influence over a process of an agent A due its attribute Att, $P_i, \ldots, P_k$ be sets parameters of the form p_{ij}^k and $i = 1, \ldots, k$. Then the* Influence Function Parameter Schema (IFPS) *of agent A upon its attribute Att during a certain period of time is the structure*

$$\delta = (Att, [func(\pi_1, n_1, [[p_{11}^1, \ldots, p_{1m}^1], \ldots, [p_{n_1 1}^1, \ldots, p_{n_1 m}^1]]),$$
$$\vdots$$

$$func(\pi_k, n_k, [[p^k_{11}, ..., p^k_{1l}], \ldots, [p^k_{n_k 1}, ..., p^k_{n_k l}]])])$$

The elements in the second argument of δ are members of a set Σ_δ, and are called the function parameter scheme (IFPs). *The third argument of function $func/3$ is called* parameter scheme *and Σ_{PSch} is a set of elements of this form. If A has n attributes then $\Delta = [\delta_1, \ldots, \delta_n]$, where each δ_i is the IFPS associated to the i–th attribute.*

This should be interpreted as *the influence upon the attribute Att is computed by using functions $\pi_1, \ldots, \pi_k$, each one with $n_1, \ldots, n_k$ parameters, respectively, and there is a list with all candidates to the first up to the last parameter for each function.* Note that here we do not care how many instances are there, and how they should be combined.

This is a dynamic structure because it represents all information relevant to the attribute of an agent during a certain period of time. Now, what we need to do is to show how to generate an agent's IFPS and then what it is used for. The construction of each element of Δ is obtained by the following function. Let Σ_{inf} be a set of protocol interface according to **Definition 5.1** and $2^{\Sigma_{par}}$ be the set of all possible domains of the functions of influence of an agent. Then the *IFPs generation* of an agent is $\Phi_\delta : \Sigma_{inf} \times 2^{\Sigma_{par}} \rightarrow \Sigma_\delta$.

The generation of an agent's IFPS is given according to the following algorithm.

Algorithm 7 IFPS Computation - $\Phi_\Delta : \Sigma_\Pi \times \Sigma_\Theta \times \Sigma_\Gamma \rightarrow \Sigma_\Delta$

Require: Σ_Π is a set of potential influences, Σ_Θ is a set of local environments, Σ_{Gamma} is a set of all information resource, and Σ_Δ is the set of IFPS of an agent, $Att \in \Lambda$ is an attribute of agent A, $\Theta_A \in \Sigma_\Theta$ is the local environment of A, $\Gamma_A \in \Sigma_\Gamma$ is the resource information acquired by A from the environment during a certain interval, $Rel \in \mathcal{L}_{env}$, $F \in \Sigma_{inf}$, Φ_γ is the accessing information resource function.

Case 1: $\Phi_\Delta([\,], \_, \_) = \varnothing$.

Case 2: $\Phi_\Delta([(\_, \_ \rightarrow Rel, F)|R], \Theta_A, \Gamma_A) = \{\Phi_\delta(F, \mathcal{S}_{\Theta_A}(Rel, \Theta))\} \cup \Phi_\Delta(R, \Theta_A, \Gamma_A)$.

Case 3: $\Phi_\Delta([(value(Att, agent(Cls)), \_, F)|R], \Theta_A, \Gamma_A) = \{\Phi_\delta(F, \Phi_\gamma(Att, Cls, \Gamma_A))\} \cup \Phi_\Delta(R, \Theta_A, \Gamma_A)$

Case 4: $\Phi_\Delta([(absorbed(amount(X, Att), agent(Cls)), \_, F)|R], \Theta_A, \Gamma_A) = \{\Phi_\delta(F, \Phi_\gamma(Att, Cls, \Gamma_A))\} \cup \Phi_\Delta(R, \Theta_A, \Gamma_A)$

Case 5: $\Phi_\Delta([(value(\_, Agent), \_, F)|R], \Theta_A, \Gamma_A) = \{\Phi_\delta(F, \varnothing\}) \cup \Phi_\Delta(R, \Theta_A, \Gamma_A)$ if $Agent$ is not a term of the form $agent(\_)$.

The next step is to translate an IFPS into a structure composed of elements which can be more easily associated to a set of FDS. Such a translation consists basically of gathering the parameters from each list in δ in a term where their position in the term corresponds to the list they belong in δ. In other words we have the following structural transformation function.

$$\begin{aligned}\delta = (Att, [&func(\pi_1, n_1, [[p^1_{11},...,p^1_{1m}], \ldots, [p^1_{n_1 1},...,p^1_{n_1 m}]]),\\ &\vdots\\ &func(\pi_k, n_k, [[p^k_{11},...,p^k_{1l}], \ldots, [p^k_{n_k 1},...,p^k_{n_k l}]])])\\ &\Downarrow\\ \nu = (Att, [&func(\pi_1, [par(p^1_{11},\ldots,p^1_{n_1 1}),\ldots,par(p^1_{1m},\ldots,p^1_{n_1 m})]),\\ &\vdots\\ &func(\pi_k, [par(p^k_{11},\ldots,p^k_{n_k 1}),\ldots,par(p^k_{1l},\ldots,p^k_{n_k l})])])\end{aligned}$$

To this second structure we call *influence function matrix* (IFM) of attribute Att, and Σ_ν to the set of all IFM of an agent. I shall refer to IFM just as the second argument of ν. The transformation function will be written as $\Phi_\nu : \Sigma_\Delta \rightarrow \Sigma_\nu$. Finally, Σ_{Par} is the set of terms of the form $par(p_1,\ldots,p_n)$. This structure should be interpreted as *the influences upon Att are obtained by using functions $\pi_1,\ldots,\pi_k$, and these functions, one at each time, have a list of instances of the parameters to be used.*

As we already know, from δ, the number of parameter instances of each function π_i we may think in terms of both structures being accessed and generated in parallel. This is possible because they are modularly defined in a matrix format and we may associate the access to the elements of one (δ) made by different threads of computation. The generation of the elements of the other one ν could also be associated to independent processes. We now show how all of this is used to generate instances of functions of influence.

The following example will help us understand these structures and the use of such functions.

Example 8 Suppose a tree t_1 has the following structures and a function scheme definition for *shadow*.

Algorithm 8 Instance of Influence Function Definition - $I_{fd} : \Sigma_\Delta \times \mathcal{F}_{inf} \rightarrow \mathcal{I}_{\mathcal{F}_{inf}}$

Require: Σ_Δ is the set of all IFPS of an agent A associated to a given attribute, $\Sigma_{\mathcal{F}_{inf}}$ is a set of all sets of influence function definitions of A, and $\mathcal{I}_{\mathcal{F}_{inf}}$ is the set of sets of instances of function definition in the form $func(n, par(x_1, \ldots, x_n), Expr)\sigma$ and σ is a set of substitutions $\{x_1/p_1, \ldots, x_n/p_n\}$ from Σ_Δ, and $PSch \in \Sigma_{PSch}$ and $Pars \in \Sigma Par$.

Case 1: $I_{fd}([\,], \_) = \varnothing$.

Case 2: $I_{fd}([func(\pi, n, PSch)|R], \mathcal{F}_{inf_A}) = \{F_{Inf}\} \cup I_{fd}(R, \mathcal{F}_{inf_A})$ if
$Pars = \Phi_\nu(func(\pi, n, PSch))$
$F_{inf} = \bigcup\{F\sigma \mid F \in \mathcal{F}_{inf_A}$ and σ is the mgu between F and $function(\pi, P, Expr)$,
where $P \in Pars$ and $Expr$ is a free variable $\}$

$$\Pi_{t_1} = [(height, [(value(position, agent(tree)),$$
$$holds(<, value(distance, value(position, agent(tree)),$$
$$value(position, t_1)), 6) \rightarrow light\_contender,$$
$$func(shadow, 2, 2)),$$
$$(value(height, agent(tree)),$$
$$holds(loc\_env\_rel(light\_contender, agent(tree), t_1),$$
$$func(shadow, 2, 1))])]$$
$$\Theta_{t_1} = [(neighbour, [(t_2, 3), (t3, 5)])]$$
$$\Gamma_{t_1} = [(height, tree, [(t_2, 9), (t_3, 8)])]$$
$$\mathcal{F}_{inf_{t_1}} = \{function(shadow, par(H, D), -(H/(D * 1000)))\}.$$
$$\Sigma_{\Delta_{t_1}} = \Phi_\Delta(\Pi_{t_1}, \Theta_{t_1}, \Gamma_{t_1}) = [(height, [func(shadow, 2, [[9, 8], [3, 5]])])]$$
$$\Sigma_{\nu_{t_1}} = \Phi_\nu(\Sigma_{\Delta_{t_1}}) = [(height, [func(shadow, [par(9, 3), par(8, 5)])])]$$
$$I_{fd}(\Sigma_{\nu_{t_1}}, \mathcal{F}_{inf_{t_1}}) = \{\ function(shadow, par(x_1, x_2), -(x_1/(x_2 * 1000)))\{x_1/9, x_2/3\},$$
$$function(shadow, par(y_1, y_2), -(y_1/(y_2 * 1000)))\{y_1/8, y_2/5\}\}$$

Because Γ_{t_1} is interactively generated, then one may choose either to generate $\Sigma_{\nu_{t_1}}$ along with Γ_{t_1} or to leave it as part of the assimilation process. There is no obvious reason why the former would be more efficient than the latter even if a real parallel architecture had been used. I took the latter choice in this work. The assimilation processes transform the resource information acquired into an attribute change factor. The attribute change factor is a structure defined as follows.

Definition 6.15 (Attribute Change Factor) *Let $Att_1, \ldots, Att_n \in \Lambda$ be a set of attributes of an agent A, and $ACF_1, \ldots, ACF_n$ be a set of (numerical, tuples, etc.)*

values of attribute change factors assimilated during a certain period of time. Then The assimilated resources of A is the structure $\Upsilon = [(Att_1, ACF_1), \ldots, (Att_n, ACF_n)]$.

The assimilation function is domain dependent, so a scheme of definition must be provided by the modeller. In the Algorithm 9, we have a scheme for assimilation assuming that for each function there exists an algebra to compute it.

Algorithm 9 Influence Assimilation Scheme - $\Re : \Sigma_\Delta \rightarrow \Sigma_\Upsilon$

Require: Σ_Δ is the set of IFPS and Σ_Υ is the set of assimilated resource, and $IFPs \in \Sigma_\delta$, $F_i \subseteq \Sigma_{\mathcal{F}_{inf}}$ with k instances of functions of influence.

Base case: $\Re([\,]) = \varnothing$.

Recursion: $\Re([(Att, IFPs)|R]) = \{(Att, V_{inf})\} \cup \Re(R)$ if

Every function scheme in $IFPs$ is defined in F_i and

$I_{fd}(IFPs, F) = F_i$

$V_{inf} = \biguplus_{j=1}^{k} V_j$ such that $eval(Expr_j, V_j)$ holds for all $Expr_j$ whose function $\pi \in F_i$.

As soon as the set of attribute change factors has been computed, then the changes on the attributes may be executed. Once again there is no general function which can be applied to every domain. Thus, a modeller must define a scheme of execution of changes based on the agent's previous state, the set of processes to execute the changes and the IAS of the agent. For this we have to assume a scheme of definition to compute changes. I shall write $compute(Proc, F)$ to say that the computation of process $Proc$ is done by F, where F is either

- a function name $\pi \in \mathcal{N}_F$, or
- an expression involving $att(A_1), \ldots, att(A_n)$ where each A_i is an attribute of the agent. This could be, for example, an arithmetic expression or any other calculus to compute one attribute in terms of others.

The other scheme definition relates the function F to compute a process to change an attribute Att associated with P, its previous value V_i and the influence assimilated Inf from the environment to the value V_j of the attribute in the next state. This will be represented by $execute(F, Att, Inf, V_i, V_j)$.

The specification of an action in **Definition 5.4** does not depend on the ordering relationship among processes. However, to specify a general scheme of execution of

processes we need to consider the temporal granular scheduler of an agent (**Definition 5.6**). This scheme of execution is implemented in the Algorithm 10.

Algorithm 10 Action - III - $\mathcal{A} : \Sigma_{\mathcal{P}_{hp}} \times \Lambda \times \Sigma_{\Upsilon} \rightarrow \Lambda$

Require: $\Sigma_{\mathcal{P}_{hp}}$ is set of sets of processes with highest priority of execution in a $\mathcal{T}_{GS}$ of an agent A, Λ is a set of possible states of A, and Σ_{Υ} is the set of assimilated resources, $P \in \mathcal{P}_{hp}$, $RP \subset \mathcal{P}_{hp}$, and $\lambda \subset \Lambda$ is A's set of states.
Base case: $\mathcal{A}([\,], \lambda, \_) = \lambda$
Recursion: $\mathcal{A}([P|RP], \lambda, \Upsilon) = \{(Att, V_j)\} \cup \mathcal{A}(RP, \lambda \setminus \{(Att, V_i)\}, \Upsilon)$ if
$(Att, V_i) \in \lambda$ such that $change(P, Att)$
$compute(P, F)$
$(Att, Inf) \in \Upsilon$ holds
$execute(F, Att, Inf, V_i, V_j)$.

6.8.5 Important Properties of an EABS

In every DAI or distributed computed architecture one has to avoid the system stop running caused by deadlock situations, and also ensure that agents have equal opportunities to perform their actions (fairness). Note that these concepts here are used in the sense of distributed computing. To avoid confusion with deadlock as mentioned in Section 4.6.1 I shall say here that the architecture does not stop running, even if the state of the world does not change, i.e. time clock is always updated by *envagent*. If a modeller specifies that under certain circumstances or state of an agent it should not change its state, then we may reach a situation in which the whole group does not change. Even in this case, agents will keep asking for *time–token* and exchanging messages. We assume that:

- the functions defined by the modeller are computable,
- the meta-relations in the set of potential influences do not contain any rule scheme which may lead to an eternal loop or perpetual process,
- the only kinds of messages agents "understand" are those specified in the protocol of communication.

I shall now give an informal proof that the meta-interpreter for Ecological Agent Based System (Eco-ABS for short) framework has the property of fairness and runs continuously. One important fact used in the proof is that information resource is $\mathcal{S}^n_{\Gamma}$, i.e.

$\Gamma_n = \mathcal{S}_\Gamma^{n-1}(\alpha_{n-1}, \Lambda, \mathcal{S}_\Gamma^{n-2}(\alpha_{n-2}, \Lambda, \ldots, \mathcal{S}_\Gamma^0(\alpha_0, \Lambda, \Gamma_0)))$.

This is not difficult to see since each $\mathcal{S}_\Gamma^i(\alpha_i, \Lambda, \Gamma_i)$ generates the next Γ^{i+1}. Note that the index of the messages α_i does not impose an order on the sensing message of the agents within the agent's local environment, it is just to associate "message slot" in the expression above with the number of expected messages. Another thing is to guarantee that this is always the case even when a given agent leaves the local environment.

Let's assume that an agent B leaves Θ_A of agent A. Suppose A waits forever for B's message, which would correspond to α_i. But according to an agent's life cycle in *Figure 6.3*, B attends all queries in $\mathcal{Q}$ and those it has not read yet. Then, B attends A query before it goes out, and A eventually reads α_i which is absurd for we assumed A waits for ever. Thus, A will not stay for ever in a waiting condition in this situation if it happens with all agents in Θ_A. Therefore A will read all answers to its sensing messages. □.

Theorem 3 (Eco-ABS is Free of Starvation) *Every ecoagent A of an Eco-ABS $\Sigma_\mathfrak{V}$ with state of computation $\Im$ will eventually be granted resource information and time–token.*

[PROOF] - *The proof is by contradiction. Assume that there is an an agent whose request will never be attended by other agent to which a query, request or inform was sent. We must consider the situations in which this might happen. Since the current Eco-ABS does not implement sensing of movements, the only situations in which this could happen are:*

1. *A multi-casts its entrance to GI and is waiting for reply while $B \in GI$ just received a command to leave. A Stays forever waiting for B, i.e. $\mathcal{S}_\Theta(GI, \Pi_A)$ of step 7 of EABS meta-interpreter (Figure 6.2) is undefined.*

   Assume this is true. If A multi-casted its entrance it has received permission from envagent (step 5) and by the envagent meta-interpreter of Section 6.7.7, A is already part of the group. By the agent's life cycle (Figure 6.3) B will attend A because either $A \in \Theta_B$ or B reads all messages addressed to it. The former case is only possible if $B \in \Theta_A$, and so A received B's reply. In the latter, B will

attend A. This contradicts the assumption.

2. *an agent is waiting for time–token and never receives it.*

   Assume this is true. This can happen only if A never reaches the top of envagent's scheduler. By the TTDP, even if A's behaviour is specified at the coarsest time scale, it will eventually reaches the top, unless the rest of the group never asks time–token again. Suppose A is part of a chain $A_1 \rightarrow \ldots \rightarrow A_n$ *of agents, where for* $i > j$ $A_i \rightarrow A_j$ *means* A_i *is waiting resource from* A_j*. According to agent's life cycle (Figure 6.3), there is no cycle in the chain for agents attend queries while they do not receive time–token. When they receive, they leave this chain in the negotiation phase by supportive attendance of others (Figure 6.4), and so at least one at time will always progress and ask for time token (Figure 6.5). As the number of agents is finite, then all will ask time token according to their time scale and A will eventually be granted with time–token. This contradicts our assumption that A waits forever.*

3. *agent A gets permission to progress and* Γ_A *is not complete. A stays forever in the negotiation phase attending others but never receiving answer.*

   Assume this is true. By the fact that information resource is $\mathcal{S}^n_\Gamma$ *this is a contradiction since all agents in* Θ_A *will eventually answer it.*

As none of the situations above are possible, there is no agent which will never be attended. Therefore, the Eco-ABS meta-interpreter is free of starvation. □

The deadlock situation we are interested is when all agents have asked for a *time–token* but no one is granted. This means that *envagent* would not have satisfied the TTDP (Section 6.7.6) and it is waiting for requests from the group and the group is waiting permission from *envagent*. The issue is what could cause such a situation and to prove that our meta-interpreter avoids it.

One possibility is that an agent receives a sensing message from a new component but does not have the right mechanisms for that class of agent. As it fails, the other agent will stay waiting forever, causing the envagent also to wait forever for its request for *time–token*. This is not a problem of the architecture, but of the modeller who might

have specified a potential influence to a class without having given the counterpart of the influence of the other one. Moreover, such a situation would not happen anyway since one agent may simply reply that a message could not be processed.

As far as the architecture proposed here is concerned I assume that deadlock could happen only in the following situations shown in the proof.

Theorem 4 (Eco-ABS Meta-Interpreter is Free of Deadlock) *The Eco-ABS meta-interpreter is free of deadlock.*

[PROOF] - *The proof is also by contradiction. Assume that env-agent never updates time–token because agents stop their interaction and do not ask for time–token. Again, we have to consider the situations in which deadlock might occur in the execution of the Eco-ABS meta-interpreter.*

1. *all agents have requested for time–token and never read a global time with the token requested and* envagent *is waiting for messages from them.*

   Assume this is true. From the Time Token Distribution Policy of Section 6.7.6, if all agents have requested time–token those on the top of the envagent scheduler will be eventually read a global time which unifies with their time–token requested. This contradicts the assumption above. Therefore it is not the case that they will be waiting for each other for ever.

2. *a group of G_1 of agents have received time–token and have incomplete Γ. They stay forever in the negotiation phase because they never receive answer for their sensing messages.*

   Assume this is true. Suppose those in a group G_2 which should answer sensing messages from G_1 are in their life-cycle phase (Figure 6.3). If G_2 has not read such messages it is because they either have been sent out or have also been granted the time–token. In the first case, G_2 go to the final attendance and then answer any query made to them, including those made by G_1. This contradicts the initial hypothesis that G_1 does not receive any answer. In the second case, those in G_2 go to change their state. If they do not sense any answer they certainly sense queries or informs, and so answer G_1 by solidary attendance.

But if they sense their answer first, they may change their state, ask time–token for the next step, and then go back to life-cycle. However, because G_1 is waiting to progress and have not asked time–token, then envagent *do not release the new time–token. Because of this those in G_2 eventually read the previous queries and the others messages, and so answer the queries from G_1. This contradicts the initial assumption G_1 does not leave the negotiation phase.*

As none of this situations above is possible, therefore the Eco-ABS meta-interpreter is free of deadlock. □.

6.9 Summary

We propose a framework in which the modeller writes her/his models for representing individuals or groups of them by using the same notation as presented in Table 6.1. Along with this, the modeller also has to define the specific domain features of the model or the definitions of the functions to compute change and influences (the modeller can take advantage of a library of such functions). Then, we offer mechanisms of execution to make such models interact, no matter their scale of time or structure.

Such mechanisms are transparent to the user's definitions, and they are implemented in a distributed symbolic architecture for multi *ecoagent* systems in which:

- an ideal medium of communication was assumed,
- agents are connected to such a medium to exchange messages (optionally, this could be more than one medium),
- agents execute one message at time and the order messages arrive is non-deterministic,
- multi-agent coordination is made by an special agent which has the function to represent properties of the group. However, *envagents* only synchronises their action in time,
- agents interact with others while they are waiting for *time–token*. During such interaction they may acquire resource or negotiate if necessary.

- we have a set of names (for class, agents, attributes, and local environmental relations), set of all possible sets of attributes (or states), a set of potential influences, a set of functions of influence upon agents' processes, a set of resource assimilation functions related to resource assimilation factor, a set of internal actions or computational functions associated to actions, a function for sensing the environment and from it generate the agent's local environment, a function for sensing changes in the agent's local environment, a function for sensing information resource, a **Naturetime** temporal reasoner.

- the meta-interpreter proposed is free of deadlock and starvation.

Chapter 7

Application of Ecoagent-Based Simulation

7.1 Introduction

In the last chapter we saw one possible way to implement an *ecoagent* based simulation architecture. In this chapter, we shall see the results of some experiments in which this architecture was applied to the modelling of ecosystems (henceforth Eco-ABS stands for Ecological Agent Based Simulation). The examples we shall see are grouped into two categories. Both have the purpose of investigating the suitability of using an Eco-ABS approach for building medium to large prototypes of simulation models with the same level of complexity of typical ecological models. The first category is concerned with this suitability in terms of the scalability of the system for large scale simulation, while the second is concerned with the expressive power and limitations of the architecture proposed in Chapter 6. In the end I shall suggest what could have been done to overcome the limitations of the present architecture.

7.2 Application on Small Scale Simulation of Ecosystem

7.2.1 Scenario One

Let's consider that the tree of **Example 1** which is actually a small portion of a forest with ten trees as depicted in Figure 7.1. I assume each tree is placed in the middle of the square it belongs, there is a cloud of *bug_demon* which moves up and down in a

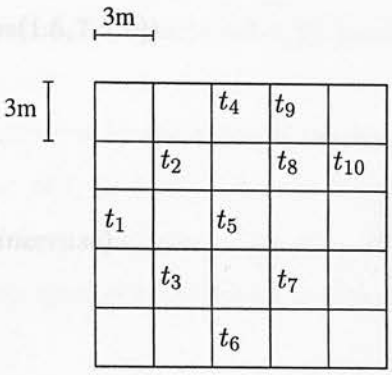

Figure 7.1: A hypothetical forest with ten trees and a cloud of bugs.

weekly scale and horizontally in a fortnight scale. No passive agent will be represented since I have not proposed any model of execution for this kind of *ecoagent*. Thus, I consider that nutrient and water uptake processes of trees are abstractly represented by a change rate of the height. This example is deliberately simple but I scale it up later.

7.2.2 An Ecoagent Model for Tree

I will not describe every single model of tree but just an *ecoagent* for one tree and another one for a bug. All other trees are very similar except they have different values for height, specific weight etc.We will first see the specifications for attributes, process-attribute relation, scale of processes and the names of the functions to compute them. Using *ecoagent* description notation we have, where *logistic* is the name the function to compute growth, *average* is the name of the function which takes a list of value and compute the average among the values.

$ecoagent(t_1, tree)$.
$attribute(external, dynamic, t_1, height)$.
$attribute(external, static, t_1, position)$.
$attribute(internal, static, t_1, specific\_weight)$.
$attribute(internal, dynamic, t_1, biomass)$.
$attribute(internal, dynamic, t_1, growth\_rate)$.

$max(height, 7)$.

$initial\_value(position, \_, pos(1.5, 7.5, 0))$.

$initial\_value(height, \_, 1)$.

$value(specific\_weight, 10)$.

$change(height, growth)$.

$change(biomass, biomass\_increase)$.

$change\_rate(height, 0.01)$.

$scale(growth, week)$.

$compute(growth, logistic)$.

$compute(biomass\_increase, att(height) * att(specific\_weight))$.

Now we have to specify the potential influences over the processes of this agent. I have said that the trees being represented in this example have a standard rate of changing height, perhaps because the amount of soil resource is uniform as time goes by. However, it does not say anything about the competition that may exist among trees. We can assume they compete for light and nutrients, and that such competition reduces their rate of change.

This influence depends on the distance among them and their height. I then abstract such a competition for light and nutrients in just one single function called *shadow* which takes height and distance as parameters. This and other functions and how they are executed are defined in Appendix C. This sort of influence among trees will be represented by a local environmental relation I call *contender*.

The other sort of influence I consider is the possible presence of bug pest. In a more complex model one might consider other kinds of classes of potential influence. This is not relevant for our purposes. The influence of bug pest (named *bug_pest*) will be computed by *pest* function which takes the value of the altitude of the cloud of bugs as parameter. This is supposed to represent that the altitude of this agent is related to the attacks it can perform on the leaves of the tree. This influence of a bug over trees is that of a *malentity* as far as the trees are concerned. We will see that from the *bug_pest* point of view this relation has other name and semantics. We then have the following representations and I explain their meaning as well.

$affect(growth, t_1, value(height, agent(tree)))$,

$holds(loc\_env\_rel(contender, agent(tree), t_1)),\ func(shadow, 2, 1)).$

The growth of agent t_1 is affected by the value of the height of an agent of class tree if such agent is a contender of t_1 and this influence is computed by function shadow which takes this height as first parameter of its two. Note that *agent of class tree* refers to any instance of this class of agent which is a contender of t_1.

$affect(growth, t_1, value(position, agent(tree)),$

$holds(<, value(distance, value(position, agent(tree)),$

$value(position, t_1)), 6) \rightarrow contender,$

$func(shadow, 2, 2)).$

The growth of agent t_1 is affected by the position of an agent of class tree if it is the case that if the value of the distance between the position of such an agent and the position of t_1 is less or equal than 6, then this agent is a contender of t_1 and the influence is computed by function shadow which takes such distance as second parameter of its two.

$affect(growth, t_1, value(altitude, agent(bug\_pest)),$

$holds(loc\_env\_rel(malentity, agent(bug\_pest), t_1)),\ func(pest, 1, 1)).$

The growth rate of agent t_1 is affected by the altitude of an agent of class bug_pest if such an agent is a malentity for t_1 and this influence is computed by pest function which takes altitude as its only parameter.

$affect(growth, t_1, value(position, agent(bug\_pest)),$

$holds(\leq, value(distance, value(position, agent(bug\_pest)),$

$value(position, t_1)), 2) \rightarrow malentity, constraint).$

The growth of agent t_1 is affected by the position of an agent of class bug_pest if it is the case that if value of the distance between the value of the position such an agent

and the value of t_1's position is less or equal than 2, then this agent is a malentity for t_1 and this is a sufficient constraint for this affect relationship.

I finally define how t_1 assimilates the value of such influences as follows.

$compute\_influence(value(height, agent(tree)), average).$

$compute\_influence(value(altitude, agent(bug\_pest)), average).$

7.2.3 An Ecoagent Model for Bug Pest

I assume that the maximum horizontal and vertical ratio of movement of a cloud of bugs being considered are represented by means of internal attributes. I name this agent b_1, and its initial position will be represented by a random function which has the measures of the sides of the forest as parameters. The meaning of the potential influences should be read in the same way as for the specification for agent t_1 above. The full description of this agent is given as follows, where *block_move* is the name of the function to compute how the height of a tree interferes in the movement of *bug_pest*; *up_zig_zag* and $flat\_zig\_zag$ are names for the functions to compute the vertical and horizontal movement of *bug_pest*, respectively; and *resource* is the local environmental relation between a an agent of class *tree* and *bug_pest* [1].

$ecoagent(b_1, bug\_pest).$

$attribute(external, dynamic, b_1, altitude).$

$attribute(external, dynamic, b_1, position).$

$attribute(internal, static, b_1, ratio_x).$

$attribute(internal, static, b_1, ratio_y).$

$max(altitude, 5).$

$value(ratio_x, 3.0).$

$value(ratio_y, 2.5).$

$initial\_value(altitude, \_, 3).$

$initial\_value(position, s(Lx, Ly), pos(X, Y, 0)) \Longleftarrow$

$random\_value(0.0, L_x, X)$ &

[1] This relation is counterpart of the *malentity* relationship between and instance of *bug_pest* and an instance of a *tree*.

$random\_value(0.0, L_y, Y)$.

$change(altitude, up\_movement)$.

$change(position, flat\_movement)$.

$scale(up\_movement, day)$.

$scale(flat\_movement, fortnight)$.

$compute(up\_movement, up\_zig\_zag)$.

$compute(flat\_movement, flat\_zig\_zag)$.

$affect(up\_movement, b_1, value(position, agent(tree))$,

$holds(<, value(distance, value(position, agent(tree))$,

$value(position, b_1)), 2) \rightarrow resource, constraint)$.

$affect(up\_movement, b_1, value(height, agent(tree))$,

$holds(loc\_env\_rel(resource, agent(tree), b_1))$, $func(block\_move, 1, 1))$.

7.2.4 An Envagent Model for Forest

I assume that the forest is fully controlled, and so its size is static. The forest is sub-divided into zones of 9m each, starting at coordinates $(0, 0, 0)$. As attributes of this *envagent* I consider the sides of the forest, the abstract position of its left bottom corner (considering the picture of Figure 7.1), the biomass of the trees which are placed within the forest, and its top-level. This last attribute is usually used as a parameter to calculate the way in which the whole forest interferes with individuals after long periods of time. But another purpose might be to know which sorts of pest could attack the canopy of the trees. I assume the last one is the only purpose of this example, but one could extend the potential influences of a tree by including this information.

$envagent(f_1, forest)$.

$attribute(external, static, f_1, side_x)$.

$attribute(external, static, f_1, side_y)$.

$attribute(external, static, f_1, position)$.

$attribute(external, static, f_1, zone\_size)$.

$attribute(external, dynamic, f_1, top\_level)$.

$attribute(external, dynamic, f_1, biomass)$.

$attribute(external, static, f_1, area)$.

$value(position, pos(0,0,0))$.

$value(side_x, 15)$.

$value(side_y, 15)$.

$value(zone\_size, 9)$

$change(top\_level, group\_growth)$.

$change(biomass, group\_biomass)$.

$compute(group\_growth, average)$.

$compute(group\_biomass, sum)$.

$affect(group\_growth, f1, value(height, agent(tree)), true, func(same\_value, 1, 1))$.

$affect(group\_biomass, f_1, value(biomass, agent(tree)), true, func(same_value, 1, 1))$.

$compute\_influence(value(height, agent(tree)), each\_one)$.

$compute\_influence(value(biomass, agent(tree)), each\_one)$.

With these specifications I ran two sorts of simulation. In the first, all classes of agents (trees and bugs) enter into the environment at the same time. In the second, I introduced them (by groups) at different times to show the non-aligned behaviour and the reaction of the system to the introduction of new agents. The conditions in which these simulations were carried out and what sort of analysis I made are explained as follows.

7.2.5 Empirical Measures of Computation Time

One of the possible advantages of the agent based architecture described in this thesis is that we can distribute parts of the computation of the model to different machines. Different forms of model will be able to gain from this to different degrees, since not all models have great potential for concurrent processing. The degree of time saving which can be achieved by distributing the agents to different machines is therefore an empirical question, which is sensitive to the choice of model. My evaluation of the computational performance of the system is therefore confined to the analysis of test runs, using a model which I consider typical of this domain.

There is also a cost to distributing the computation, since the machines containing the different agents must communicate and communication delays cost time. In other

words, the normal measure of CPU time on each machine would not give an accurate picture of the cost for the system as a whole. For this reason I kept information about the time each agent had to compute its state transition plus the system time plus communication time. This is usually known as wall clock time. This measure can be problematic because of possible variations in the speed of machines being used due to the number of processes competing for their resources. In order to reduce such variations, the results we shall see here are from simulations executed during the same period of time, at different days when most of the machines were not been used. However, we still have the possibility that the usual processes of file system management (e.g. backups, email servers, etc.) have exerted some influence on these measures.

In order to simulate different modellers integrating their models in a distributed environment each agent, including the *envagent*, was installed on a separate machine (each of these being a client to a central server). For each one a file was generated as a log of the history of the simulation. I next present the outcomes of these simulations.

7.2.6 Simulating Behaviour Aligned in Time

In Table 7.1 we show the evolution of *bug_demon*'s state until $t(15, 1, 1)$ where its slowest process changed its position. Note that its local environment (Θ) is empty because no other agent satisfies the conditions described in its potential influences to be considered as actual influences.

Figure 7.2 shows the running of the bug where the axis x represents the states of it in days, the axis y represents the waiting time plus executions time plus time for communication or interaction with other agents. This allows us to observe the behaviour of the system when I scale it.

The higher peaks (A) occur because of the extra effort needed to change the bug's position at weekly scale. The peaks (B) close to the x axis point out the time that the up and down movement of the bug takes to be executed. Those higher peaks indicate the bug had these processes aligned in time with the process of tree's growth which is slower than the bug's movement. Because the trees had to change their state, we then

State	Time	Attributes	Θ
0	$t(1,1,1)$	$[(altitude, 3), (position, pos(34.3, 17.95, 0))]$	$\varnothing$
1	$t(2,1,1)$	$[(altitude, 1.84), (position, pos(34.3, 17.95, 0))]$	$\varnothing$
2	$t(3,1,1)$	$[(altitude, 0.82), (position, pos(34.3, 17.95, 0))]$	$\varnothing$
3	$t(4,1,1)$	$[(altitude, 0.08), (position, pos(34.3, 17.95, 0))]$	$\varnothing$
4	$t(5,1,1)$	$[(altitude, 0.91), (position, pos(34.3, 17.95, 0))]$	$\varnothing$
5	$t(6,1,1)$	$[(altitude, 0), (position, pos(34.3, 17.95, 0))]$	$\varnothing$
6	$t(7,1,1)$	$[(altitude, 1), (position, pos(34.3, 17.95, 0))]$	$\varnothing$
7	$t(8,1,1)$	$[(altitude, 0.08), (position, pos(34.3, 17.95, 0))]$	$\varnothing$
8	$t(9,1,1)$	$[(altitude, 0), (position, pos(34.3, 17.95, 0))]$	$\varnothing$
9	$t(10,1,1)$	$[(altitude, 0.12), (position, pos(34.3, 17.95, 0))]$	$\varnothing$
10	$t(11,1,1)$	$[(altitude, 0), (position, pos(34.3, 17.95, 0))]$	$\varnothing$
11	$t(12,1,1)$	$[(altitude, 0.17), (position, pos(34.3, 17.95, 0))]$	$\varnothing$
12	$t(13,1,1)$	$[(altitude, 0), (position, pos(34.3, 17.95, 0))]$	$\varnothing$
13	$t(14,1,1)$	$[(altitude, 0.22), (position, pos(34.3, 17.95, 0))]$	$\varnothing$
14	$t(15,1,1)$	$[(altitude, 0), (position, pos(35.01, 18.78, 0))]$	$\varnothing$

Table 7.1: State transition table of *bug_pest*.

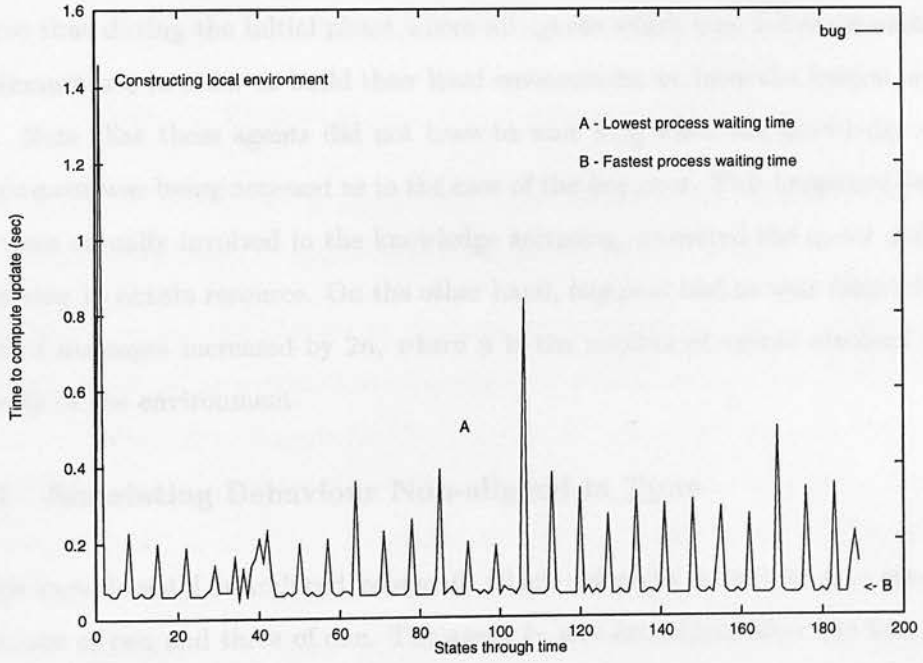

Figure 7.2: Graphic of wall-time to change state through the flow of logical time.

had a higher traffic of message which explains why the bug has to wait more than in the other cases.

The highest peak of all indicates the time this agent had to wait because the knowledge about some property of the environment was being accessed. This was done by sending the following query, where $solve/3$ is an extension of $solve/3$ seen before to deal with queries for the distributed version of the system. There are two additional variables: T_1 represents the wall time for the answer given and T_2 represents the wall time measured to wait the final answer sent by the *envagent*.

$solve(value(top\_level)\ @\ t(29,1,1), A_1, T_1)\ \&\ solve(value(biomass)\ @\ t(29,1,1), A_2, T_2)$.

- $A_1 = value(top\_level, 1.04)\ @\ t(29,1,1)$
- $A_2 = value(biomass, 103.08)\ @\ t(29,1,1)$
- $T_1 = 227.82s$
- $T_2 = 228.27s$

For this example the local environment of each tree is static, and so I do not show it here. Figure 7.3 shows the time the state progress of each tree had to wait. We observe that during the initial phase where all agents which may influence each have to communicate in order to build their local environment we have the longest waiting time. Note that these agents did not have to wait long when the knowledge of the environment was being accessed as in the case of the *bug_pest*. This happened because they were actually involved in the knowledge accessing, answered the query and kept interacting to obtain resource. On the other hand, *bug_pest* had to wait longer for the traffic of messages increased by $2n$, where n is the number of agents involved in the property of the environment.

7.2.7 Simulating Behaviour Non-aligned in Time

In this experiment I introduced *ecoagents* which represent trees into two groups of three, one of two and three of one. The agent b_1 was introduced after the first group of three. Because it behaves faster than the others when I introduced the rest they enter non-aligned in time in relation to the ones running at the time of entrance. The agents entered into the environment in the following order: $G_1 = \{t_1, t_6, t_{10}\}$, b_1, t_4,

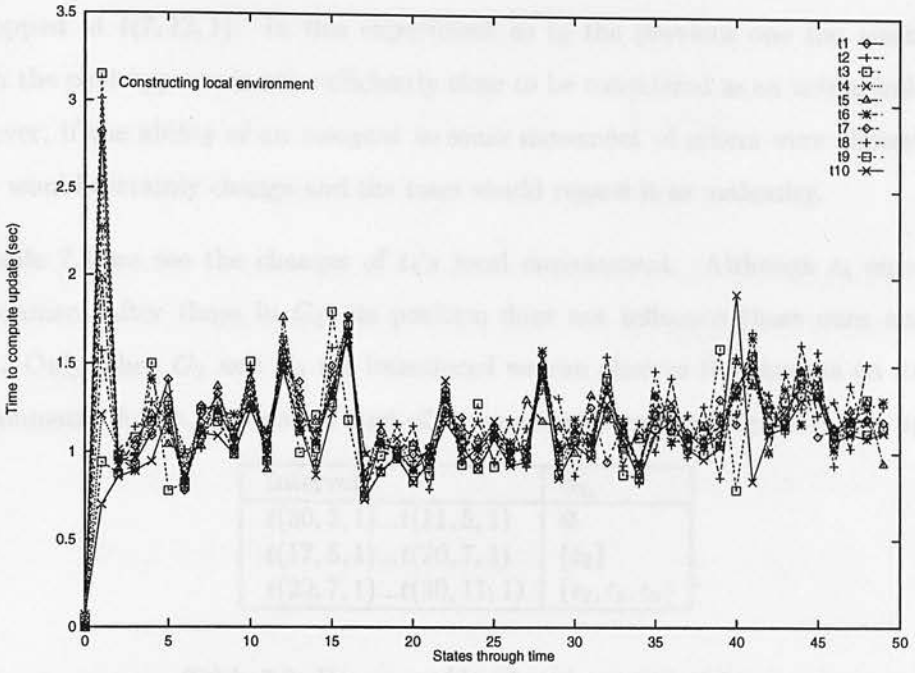

Figure 7.3: Graphic of wall-time to change state through the flow of logical time.

Interval	Θ_{t_1}	Θ_{t_6}	$\Theta_{t_{10}}$
$t(1,1,1)...t(16,5,1)$	$\varnothing$	$\varnothing$	$\varnothing$
$t(17,1,1)...t(18,1,1)$	$\{t_2\}$	$\varnothing$	$\varnothing$
$t(18,5,1)...t(16,7,1)$	$\{t_2\}$	$\{t_7\}$	$\varnothing$
$t(20,7,1)...t(3,12,1)$	$\{t_2,t_3\}$	$\{t_3,t_7\}$	$\{t_8,t_9\}$

Table 7.2: Progress of local environment of group $\{t_1, t_6, t_{t_{10}}\}$.

t_5, $G_2 = \{t_2, t_7\}$ and $G_3 = \{t_3, t_8, t_9\}$. The important thing to observe is the change in the local environment of agents as new components are introduced.

In Table 7.2 we see the changes in the local environment of G_1. I recall that the behaviour of all of its members is observed in a weekly basis. Note that the time $t(17,1,1)$ at which Θ_{t_1} changes for the first time is in between two consecutive time steps of its growth process, i.e. $t(15,1,1)$ and $t(22,1,1)$. A similar non-aligned entrance also happens with t_6 at $t(18,1,1)$. Later, at $t(20,1,1)$, both t_1 and t_6 have another change in their local environment at another time non-aligned with their update, i.e. $t(16,7,1)$ and $t(23,7,1)$. Before this event Θ_1 and Θ_6 had no intersection, and after this t_1 and t_6 interfere, indirectly, with one another's behaviour.

The local environment of b_1 remains empty from the beginning until the simulation

is stopped at $t(7,12,1)$. In this experiment as in the previous one the position in which the *pest* appears is not sufficiently close to be considered as an actual influence. However, if the ability of an *ecoagent* to sense movement of others were allowed then its Θ would certainly change and the trees would regard it as malentity.

In Table 7.3 we see the changes of t_4's local environment. Although t_4 enters the environment after those in G_1, its position does not influence those ones and vice versa. Only when G_2 and G_3 are introduced we can observe the changes on its local environment. Again, t_2 becomes part of Θ_4 in a time step non-aligned to t_4's update.

Interval	Θ_{t_4}
$t(30,3,1)...t(11,5,1)$	$\varnothing$
$t(17,5,1)...t(20,7,1)$	$\{t_2\}$
$t(20,7,1)...t(30,11,1)$	$\{t_2, t_8, t_9\}$

Table 7.3: Progress of local environment of t_4.

Table 7.4 shows the progress of t_5, and we can see it is the agent which is more affected by the introduction of new components. It is not a surprise since it is placed in the middle of the area of the forest (see Figure 7.1 of Section 7.2.1). Note that Θ_{t_5} changes twice in between the time steps $t(13,5,1)$ and $t(20,5,1)$ because of t_2 and t_7's entrance, respectively.

The agents of G_2 also enter at different time as we can see in Table 7.4 above. Their local environment are shown in Table 7.5.

Table 7.6 shows that the last group have their local environment unchanged until the computation of the whole system is shut down. One important thing to notice is that I have presented how *ecoagents* are sensible to changes in the environment and by using their potential influences they quickly respond to such changes. It is worth saying

Interval	Θ_{t_5}
$t(1,4,1)...t(13,5,1)$	$\varnothing$
$t(17,5,1)...t(18,5,1)$	$\{t_2\}$
$t(18,5,1)...t(15,7,1)$	$\{t_2, t_7\}$
$t(20,7,1)...t(2,12,1)$	$\{t_2, t_3, t_7, t_8\}$

Table 7.4: Progress of local environment of t_5.

Interval	Θ_{t_2}	Θ_{t_7}
$t(17,5,1)...t(6,12,1)$	$\{t_1, t_5, t_4\}$	$\{t_6, t_5\}$
$t(6,12,1)...t(7,12,1)$	-	$\{t_6, t_5\}$

Table 7.5: Progress of local environment of group $\{t_2, t_7\}$.

that in cases similar to this example, where we have a non-aligned interaction among agents, I adopted the policy presented in Section 4.7.2 for the sequential computation in **NatureTime**.

Interval	Θ_{t_3}	Θ_{t_8}	Θ_{t_9}
$t(20,7,1)...t(7,12,1)$	$\{t_6, t_1, t_5\}$	$\{t_9, t_5, t_4, t_{10}\}$	$\{t_4, t_{10}, , t_8\}$

Table 7.6: Progress of local environment of group $\{t_3, t_8, t_9\}$.

7.3 Experiments with Large Scale Ecological Models

7.3.1 Scenario Two

I now extend the example of Section 7.2.1 with one hundred trees and the bug now also in large numbers. Figure 7.4 shows the new example where I simply considered a larger area from the previous example and introduced more trees. For this new example, the values for $side_x$ and $side_y$ change both to 45, but I keep the same size for each zone which will give us 25 zones.

The representation of the agents follows the same pattern of Sections 7.2.2, 7.2.3 and 7.2.4. I take those descriptions except the one for the forest which changes some of its attributes, and also assume all other trees have same set of potential influences, assimilation functions, etc.One might consider it relevant to model different classes of trees with different features, but this is a modelling issue which does not raise any interesting question for the purpose of analysing the architecture proposed.

I ran two classes of simulation. In the first I put all agents into the environment at the beginning. In this I could evaluate the worst case of message exchange which is when all agents have to build their local environment at the same time. During this simulation I moved some agents out of the environment to show how the system responded to such events. In the second class I ran models varying the number of agents in order to show

3m

3m

t_{22}		t_{23}			t_{33}		t_{41}		t_{54}		t_{71}		t_{88}	
			t_{24}		t_{34}					t_{62}		t_{81}		t_{95}
	t_{20}	t_{21}		t_{26}				t_{47}	t_{55}		t_{72}		t_{89}	
t_{19}			t_{25}	t_{27}		t_{39}			t_{56}			t_{82}		
t_{17}		t_{18}						t_{48}		t_{63}	t_{73}		t_{90}	
	t_{16}			t_{28}	t_{35}		t_{42}		t_{57}		t_{74}	t_{83}		t_{96}
			t_{15}			t_{40}			t_{58}	t_{64}	t_{75}			
	t_{14}	t_{13}		t_{29}	t_{36}			t_{49}		t_{65}		t_{84}	t_{91}	
		t_{12}					t_{43}	t_{50}	t_{66}		t_{76}			t_{97}
t_{11}				t_{30}				t_{51}		t_{67}		t_{85}	t_{92}	
		t_4	t_9		t_{37}		t_{44}		t_{59}		t_{77}			t_{98}
	t_2		t_8	t_{10}			t_{45}			t_{68}	t_{78}	t_{86}		t_{99}
t_1		t_5		t_{31}				t_{52}	t_{60}	t_{69}	t_{79}		t_{93}	
	t_3		t_7		t_{38}		t_{46}		t_{61}		t_{80}			t_{100}
		t_6		t_{32}				t_{53}		t_{70}		t_{87}	t_{94}	

Figure 7.4: A hypothetical forest with one hundred trees.

the scaling up of the system when we move from small scale to large scale simulation.

7.3.2 Simulation of Large Scale Problem

I did not access the property of the environment in this example, but made another sort of action. I moved groups of trees out of the environment. Figure 7.5 shows the progress of the *bug_pest* for this new example. The waiting time for local environment generation has increased considerably if compared to the time showed in Figure 7.2. The increase is proportional to the number of agents as we shall see in Section 7.3.3. This also happens during the progress of the agent's state, although in the case of the faster processes the difference is not so big in considering the number of agents running.

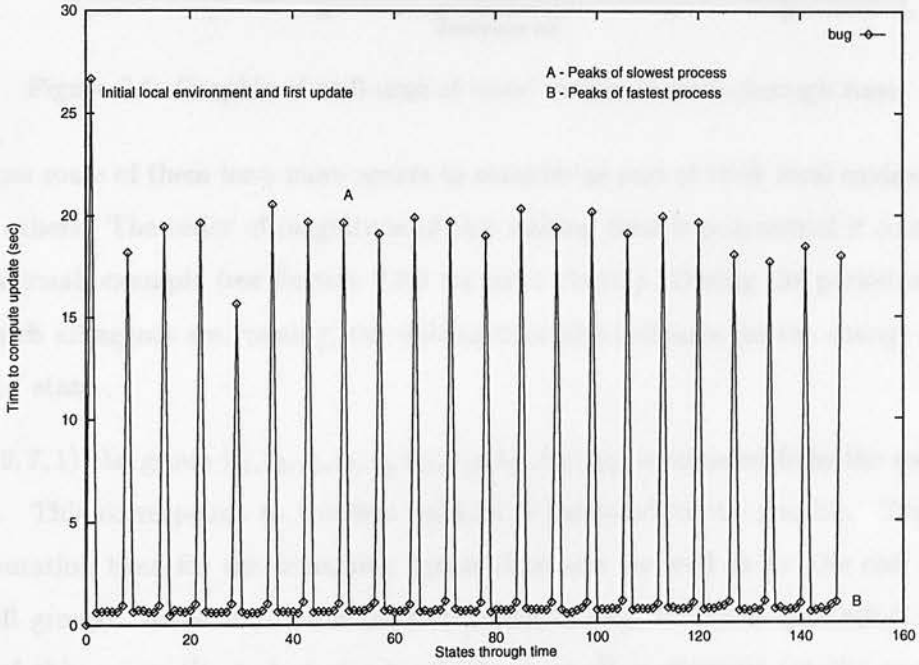

Figure 7.5: Graphic of wall-time to bug's update its state. A - corresponds to bug's slower process to change position; B - bug's faster process to change altitude.

We will see now the progress through time of some trees just to give an idea of the behaviour of the system. In Figure 7.6 we have the progress of the trees from t_{41} up to t_{60}. Note that the maximum time these agents have to build their local environment is almost three times more than *bug_pest*, and the minimum is within the same range of change, i.e. between 20 to 30 seconds. This difference of time among this group exists

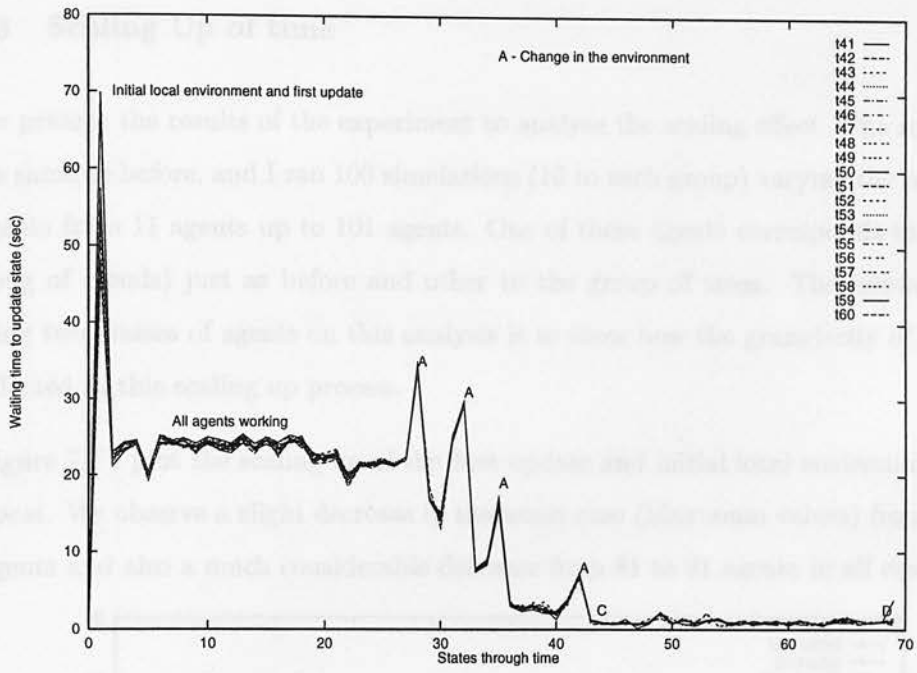

Figure 7.6: Graphic of wall-time of trees' change of state through time.

because some of them have more agents to consider as part of their local environment than others. The order of magnitude of this waiting time is polynomial if compared to the small example (see Section 7.3.3 for more details). During the period of time in which all agents are running, the waiting time also increases for the change in the agents' state.

At $t(9,7,1)$ the group $[t_1, t_3, t_5, t_7, t_9, t_{11}, t_{13}, t_{15}, t_{17}, t_{19}]$ is removed from the environment. This corresponds to the first peak at A indicated in the graphic. Then the computation time for the remaining agents decreases (as well as for the rest of the overall group). Some steps later (corresponding to four weeks) at time $t(6,8,1)$ the rest of this group ($[t_2, t_4, t_6, t_8, t_{10}, t_{12}, t_{14}, t_{16}, t_{18}, t_{20}]$) is removed (at the second A peak). At the third A peak, group $[t_{21}, t_{24}, t_{27}, t_{30}, t_{33}, t_{36}, t_{39}, t_{42}, t_{45}, t_{48}]$ is also moved out. Then, at the fourth A peak another group is moved out.

The phase from the marks C to D corresponds to a number of agents around ten. If we look at the graphic of Figure 7.3 we can see that this phase has the same average of waiting time as the agents of that figure. But this is better explained in the following section.

7.3.3 Scaling Up of time

I now present the results of the experiment to analyse the scaling effect. The scenario is the same as before, and I ran 100 simulations (10 to each group) varying the number of agents from 11 agents up to 101 agents. One of these agents corresponds to a bug (or bug of clouds) just as before and other to the group of trees. The relevance of putting two classes of agents on this analysis is to show how the granularity of action is reflected in this scaling up process.

In Figure 7.7 I plot the scaling up of the first update and initial local environment for *bug_pest*. We observe a slight decrease in the worst case (Maximum values) from 31 to 41 agents and also a much considerable decrease from 81 to 91 agents in all cases.

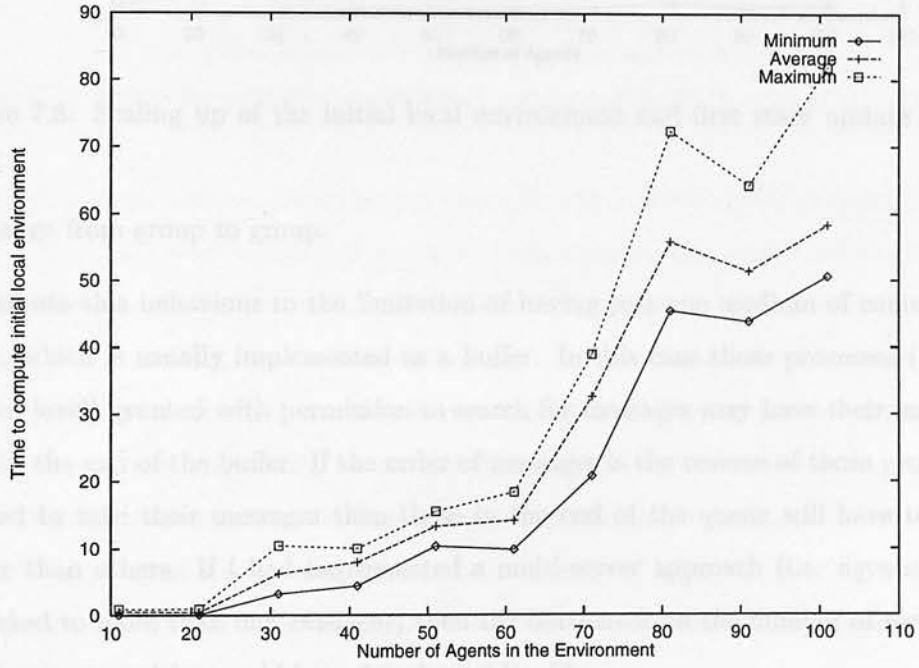

Figure 7.7: Scaling up of the initial local environment and first state update of *bug_pest* considering the total number of agents.

A similar behaviour is observed in Figure 7.8 for waiting time of the group of trees. There is an interesting variation between the best case (minimum waiting time) and the worst case (maximum waiting time). There are some agents that have a very short waiting time to generate their initial local environment no matter the number of agents. Although this can be related to the number of agents per zone of influence, it does not seem to be the real reason because this number does not have a great range

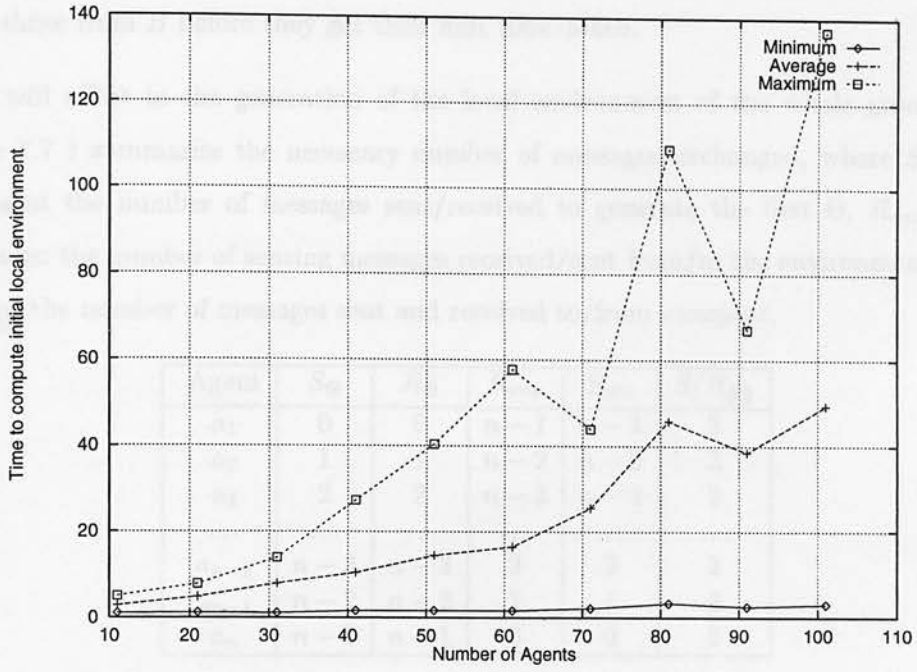

Figure 7.8: Scaling up of the initial local environment and first state update of the trees.

of change from group to group.

I attribute this behaviour to the limitation of having just one medium of communication, which is usually implemented as a buffer. In this case those processes (at the system level) granted with permission to search for messages may have their message only in the end of the buffer. If the order of messages is the reverse of those processes queued to take their messages then those in the end of the queue will have to wait longer than others. If I had implemented a multi-server approach (i.e. agents could be linked to more than one *envagent*) then the bottleneck on the number of messages and processes waiting could be reduced considerably.

Let's consider that there are n agents in a group which can be within the same net of influence as soon as they enter into the environment, assuming they all enter at the same time. Those which have first access to *envagent*'s knowledge about the current agents first will have less messages to multi-cast and to receive than those accessing later. Let's call A the first group and B the latter. Thus, those from A will need less time to generate their first local environment (Θ) than those from B. As they all enter at the same time, those from A will end up exchanging the same amount of messages

with those from B before they get their first *time-token*.

This will affect in the generation of the local environment of the whole group. In Table 7.7 I summarise the necessary number of messages exchanged, where S_Θ/R_Θ represent the number of messages sent/received to generate the first Θ, R_{env}/S_{env} represent the number of sensing messages received/sent from/to the environment, and $S/R_{\textcircled{U}}$ the number of messages sent and received to/from *envagent*.

Agent	S_Θ	R_Θ	R_{env}	S_{env}	$S/R_{\textcircled{U}}$
a_1	0	0	$n-1$	$n-1$	2
a_2	1	1	$n-2$	$n-2$	2
a_3	2	2	$n-3$	$n-3$	2
...	...	...	...	...	...
a_{n-2}	$n-3$	$n-3$	2	2	2
a_{n-1}	$n-2$	$n-2$	1	1	2
a_n	$n-1$	$n-1$	0	0	2

Table 7.7: Necessary messages to exchange among *ecoagents* in zone with n agents.

The sum of each column, from the 2nd until the 5th, is $\frac{(1+(n-1))\times(n-1)}{2}$, and the last is $2n$. Thus, the total number of messages exchanged by this group before they are in condition to receive permission to progress in the logical time is then $2n + 4 \times \frac{(1+(n-1))\times(n-1)}{2}$, or $2n^2$. As the zones of influence for each agent of the example above do not increase so drastically, and it is not even so big, even for those agents in the intersection of the geometrical zones, then the only reason for such a wide variation in the waiting time must be the limitation in the medium as proposed above. Note that, in the case of having multiple mediums associated with *sub-envagents* of each geometrical (or geographical) zone, then the numbers presented in this table give an idea of the number of messages exchanged per group.

One important thing to note is that after generating the local environment, the number of messages each agent has to exchange is even smaller than in the beginning. Of course that this number may change as time of simulation flows according to the dynamics of the system being modelled.

Finally, I present the scaling up for updating state after each tree has built its initial local environment. As in the previous graphics, we have the minimum, the average and the maximum number of messages exchanged. This is depicted in Figure 7.9 where we

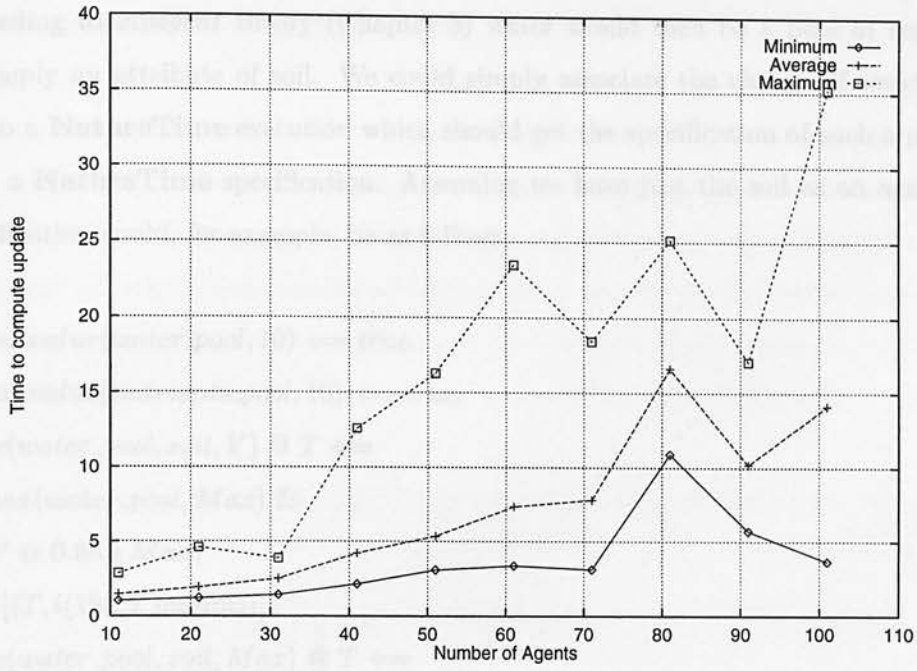

Figure 7.9: Scaling up for updating state of the trees.

can see the same pattern of decrease in the wall-time and then increase again.

The results we have seen in this section show that the bottleneck for large scale simulation of an Eco-ABS system is on the number of messages exchanged when the number of agents increases. This result is not a surprise since the price one has to pay for distributing computation is usually on the this issue. However, some improvements are possible as I have already suggested above.

7.4 Expressive Power of Eco-ABS Approach

Eco-ABS approach is expressive as **NatureTime** for representing and executing specification of interacting simulation models at various scales of time. The mechanisms for interacting agent have the same semantics of those proposed in Section 4.7.3. A friendly user interface can be used to specify classes of *ecoagents* which would then be translated to the notation of the Eco-ABS framework.

Although I have not demonstrated how to represent cyclical process in Eco-ABS, very few changes are necessary to deal with it. Suppose we want to represent the cyclical behaviour of a water in the soil of the forest represented in the examples above.

According to *ecoagent* theory (Chapter 5) water would then be a pool of resource, or simply an attribute of soil. We could simply associate the change of water storage to a **NatureTime** execution which should get the specification of such a process from a **NatureTime** specification. Assuming we have just the soil as an agent, its specification could, for example, be as follows.

$initial\_value(water\_pool, 10) \Leftarrow true.$

$initial\_value(nutrients\_pool, 10) \Leftarrow true.$

$value(water\_pool, soil, V)\ @\ T \Leftarrow$

$max(water\_pool, Max)\ \&$

$(V\ is\ 0.65 * Max)$

$::\ [(T, i(12...2, month))].$

$value(water\_pool, soil, Max)\ @\ T \Leftarrow$

$max(water\_pool, Max)$

$::\ [(T, i(3...5, month))].$

$value(water\_pool, soil, V)\ @\ T \Leftarrow$

$max(water\_pool, Max)\ \&$

$(V\ is\ 0.4 * Max)$

$::\ [(T, i(6...11, month))].$

$change(water\_pool, water\_dynamics).$

$compute(water\_dynamics, nature\_time).$

Of course some minor changes in the execution mechanisms for change on the attributes of agents should be required, mainly when passive agents have users (or other agents) for its pool of resources.

The structures proposed to represent local environment, potential sources of influence and resources to be acquired (Γ) allow changes in a running Eco-ABS application without the need to drastic changes at the code or specification level. This means that any agent can interact opportunistically with others through potential influences, so that the specific routes for communication need not to be pre-programmed.

7.5 Limitations of Eco-ABS

This approach has some limitations in terms of representation. I now discuss the most important. The first limitation is related to the third possible situation from Section 6.8.2 in which an agent's local environment (Θ) may change. This is when the agent has to sense others moving around or those moving around sense those which may become part of their own environment. A solution to this limitation could be, for example, as follows, where A is the moving agent and Θ_A its current local environment.

1. A gets the agents in Θ_A which may have their local relations with A changed, call it Θ_{A_i},
2. A asks *envagent* about which agents are within its field of influence if it moves to the desired location. Call it Θ_{A_v}.
3. A multi-casts to Θ_{A_i} it intends to move and to Θ_{A_v} it intends to be a new agent in their local environment,
4. A waits for reply from $\Theta_{A_i} \cup \Theta_{A_v}$ to build its new Θ_A or negotiate its movement if it is the case.
5. A informs *envagent* of the result of its movement.

The generality of this specification stops in the point where negotiation may be necessary. This is because the sort of negotiation may vary from domain to domain, and one should care about possible deadlocks that such a negotiation could face. This issue, though, is not relevant for the purpose of this work. To describe a detailed specification of it would increase the sophistication of the architecture, so this is left for further work.

A second limitation is that there is no mechanisms to perform the simulation of sudden changes over an agent's attribute. For example, suppose that instead of simply cutting a tree off one wants to simulate the action of cutting the canopy of the tree or just half of its height. This action could be represented as $cut(height, X)$ where X is the measure of height to be cut off. This might be problematic if the action takes place

in a time moment in between two consecutive time steps of the process in question as depicted in Figure 7.10, where $(H + \Delta_H)$ represents how the change would happen without the action.

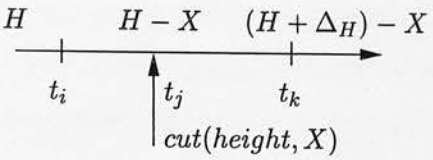

Figure 7.10: An action suddenly changing the height H between t_i and t_k.

The only problem here, if compared to the non-aligned inference examples we saw in Section 4.7.2, is that we need to extend the constraint an agent has to change its attribute. Without such sudden events, the execution of processes to change attributes depends on the *time–token* the agent needs to read from the *envagent*. The extension is then basically to allow an agent to reset on the attribute affected when it receives a message like the one above. Also, one has to care about the consequences such changes may have on other abilities of the agent. For example, a tree cannot absorb light "properly" if its canopy has been damaged. As a consequence, the processing of messages associated with the actions an agent is able to recognise from the external world needs some changes to this kind of event. However, I avoid the specification of such mechanisms for this involve a much deeper investigation on other matters that are natural extensions of this research.

How expressive is Eco-ABS approach if one wants to simulate a situation similar to the one proposed in **Example 1** of Section 1.1 ? A considerable good representation of what is going on in that example should consider the following agents, events and actions as shown in Table 7.5. In this table we have two classes of reactive *ecoagent* (*tree* and *pest*), two passive (soil of the forest which is not mentioned and p_j) and five active. It has information which could be represented by using the current architecture, things which would be difficult to represent and things which we could not represent. Representation of the four time scales is easy since we would need only to specify a time hierarchy with the four scales given, assuming that we ignore irregularities in number of days for month and leap years.

Agent	Event	Action/Process Reason	Scale
John	Meet *Elza* Meet *Minor*	Plant *tree* Attack pest with P_j	
tree	Enter into environment Attacked by pest	**growth** main goal **absorbs** p_j	week day hour
pest		Enter into environment **attacks** *tree* need food Run away from p_j Come back to attack	 day month
p_j	Used against *pest*		month
Elza	Meet *John*	**suggests** *John* to use p_b less harmful than p_j	
Minor	Meet John	**suggests** *John* to use $p_j + p_b$ keep products and jobs	
Angelina	meet John	**suggests** *John* to use *bug_angel* attack *pest* and fertilises soil	

Table 7.8: Table of Agents, events, actions and correspondent scale to be considered when modelling **Example 1.**

Among things that could be represented we can include the absorption of chemical p_j by trees. We would just need to include canopy or average area of leaves, introduce chemical resources as potential source of influence along with the functions to compute such an influence, and how this would affect the growth of the tree or, if it is the case, how this would affect seed or fruit production (where amount of seed and fruit production would be time dependent attributes of an *ecoagent* tree). Things like plant tree, attack pest with p_j and suggestions made by some agents are would be represented by the messages they send to the objects of their actions.

The most difficult things to represent using the current architecture are

- the autonomous running away and coming back behaviour of *pest* since I did not provide mechanisms for sensing movement as explained above,
- the characterisation of active *ecoagents* by using the simple framework of specification, *e.g.* how can we represent the knowledge *Angelina* has about the effects of using a certain kind of natural resource? In other words, we are not able to represent knowledge about causal relations between certain actions and the effects of them on other agents.
- the autonomous dialog among active *ecoagents* would need more sophisticated structures and mechanisms for representing dialog and negotiation to deal with the kinds of information involved.

7.6 Discussion

As I pointed in the end of Section 7.3.2 the bottleneck of the system proposed is on the number of messages. The architecture is constrained to just one medium of communication. An alternative is to extend this one to allow agents to be linked to more than one medium, and sub-divide an *envagent* into sub-groups where each one would be responsible for coordinating messages of its members only. This could be done by making use of the already partitioned topology of the environment and associating a *sub-envagent* to each zone, in which case a more elaborated coordinating algorithm would be necessary.

Another measure would be to separate the external knowledge into another medium of communication since agents would be allowed to be connected to more than one medium. The good side effect of this would be the straight use of this medium for building a friendly user interface on the top of the specification of classes of *ecoagents* and execution of models.

Despite the considerable increase in the time of computation, the results are extremely promising if compared to previous approaches using standard computational logic, for example [Robertson *et al.* 91, Mota 94]. [Robertson *et al.* 91] showed how inefficient is the simulation of models expressed in a standard Prolog style with normal execution strategy and also with meta-interpreters which cache intermediate results. In both cases no specialised guided search was proposed to consider interacting agents working at different time scales.

In this work I have proposed an special search algorithm to overcome this problem (Chapter 4). But even in this case the exponential explosion of the search space makes simulation of many interacting models a problem. The problem araises because of dependencies between various parts of the model. To make clear what the problem is I shall consider the **Example 7** and use $\mathcal{R}_1$, $\mathcal{R}_2$, and $\mathcal{R}_3$ to represent the function which compute the values of the height of t_1, t_2 and b_1 at a previous time according to their scale, and $\mathcal{F}_{prog}$ the function which composes a sequence of values. This is just to wrap the body of a simulation clause and show just what matters on this analysis. For example,

$$
\begin{aligned}
value(height, t_1) \ @ \ p(1, week) \ after \ T = \mathcal{R}_1(\ & value(height, t_1) \ @ \ T, \\
& progress(value(pos, b_1) \ @ \ T, p(1, week)), \\
& progress(value(height, t_2) \ @ \ T, p(1, week)))
\end{aligned}
$$

The innermost expressions of the indentation mean that they are the derivation of the computation of the expressions above. We already know that *progress*/3 (defined in Section 4.7.3) wraps a sequence of calls to *solve*/3 according to the aligned and non-aligned search strategy. This means that for a query $value(height, t1) \ @ \ t(13, 1, 1)$ we have

$$value(height, t_1)@t(13,1,1) = \mathcal{R}_1(\ value(height, t_1)@t(6,1,1),$$
$$progress(value(pos, b_1)@t(6,1,1), p(1, week)),$$
$$progress(value(height, t_2)@t(6,1,1), p(1, week)))$$
$$value(height, t_1)@t(6,1,1) = value(height, t_1)@t(1,1,1)$$
$$progress(value(pos, b_1)@t(6,1,1), p(1, week))$$
$$= \mathcal{F}_{prog}(value(pos, b_1)@t(6,1,1),\ value(pos, b_1)@t(7,1,1),$$
$$value(pos, b_1)@t(8,1,1), \ldots, value(pos, b_1)@t(12,1,1))$$
$$value(pos, b_1)@t(6,1,1) = \mathcal{R}_3(value(pos, b_1)@t(5,1,1),$$
$$value(height, t_1)@t(5,1,1))$$
$$value(pos, b_1)\ @\ t(5,1,1) = \mathcal{R}_3(value(pos, b_1)@t(4,1,1),$$
$$value(height, t_1)@t(4,1,1))$$
$$value(pos, b_1)@t(4,1,1) = \mathcal{R}_3(value(pos, b_1)@t(3,1,1),$$
$$value(height, t_1)@t(3,1,1))$$
$$value(pos, b_1)@t(3,1,1) = \mathcal{R}_3(value(pos, b_1)@t(2,1,1),$$
$$value(height, t_1)@t(2,1,1))$$
$$value(pos, b_1)@t(2,1,1) = \mathcal{R}_3(value(pos, b_1)@t(1,1,1),$$
$$value(height, t_1)@t(1,1,1))$$
$$value(pos, b_1)@t(7,1,1) = \mathcal{R}_3(value(pos, b_1)@t(6,1,1),$$
$$value(height, t_1)@t(6,1,1))$$
$$\ldots$$
$$value(pos, b_1)\ @\ t(12,1,1) = \mathcal{R}_3(value(pos, b_1)@t(11,1,1),$$
$$value(height, t_1)@t(11,1,1))$$
$$\ldots$$
$$progress(value(height, t_2)@t(6,1,1), p(1, week)))$$
$$= \mathcal{F}_{prog}(value(height, t_2)@t(6,1,1),\ value(height, t_2)@t(12,1,1),$$
$$value(height, t_2)@t(6,1,1) = \text{non existent}$$
$$value(height, t_2)@t(12,1,1) = \mathcal{R}_2(value(height, t_2, 10))$$

Notice that computations to obtain $value(height, t_1)@t(13,1,1)$ are very similar to those required to compute $value(height, t_1)@t(10,1,1)$ because the non-aligned search strategy will require the search to start from $t(3,1,1)$ until $t(10,1,1)$ for values of height of t_2 and of position of b_1. In this case all computations for $value(pos, b_1)@t(10,1,1)$

will be similar to those above. [Robertson *et al.* 91] made an analysis when there are just two models interacting pointing that a number of times a value can be calculated can reach the order of 10^{13} times, when the branching rate of the search tree increases to 5 and the number of levels needed to achieve the goal is 20. A solution to this problem, as they suggested, is to cache the results of quantities which are physically impossible to have more than one value at the same time. Such results were called lemmas.

I have implemented a version of **NatureTime** meta-interpreter of Chapter 4 caching lemmas. In Figure 7.11 we see a plot of the simulation of a query $value(height, t_1, H)@T$ where T is *t-free*. The time steps were generated by forcing backtracking so that a new instance of a simulation clause is generated taking the previous value into account. It was not possible to run models with more than 3 agents interacting because of the price we have to pay when we cache results.

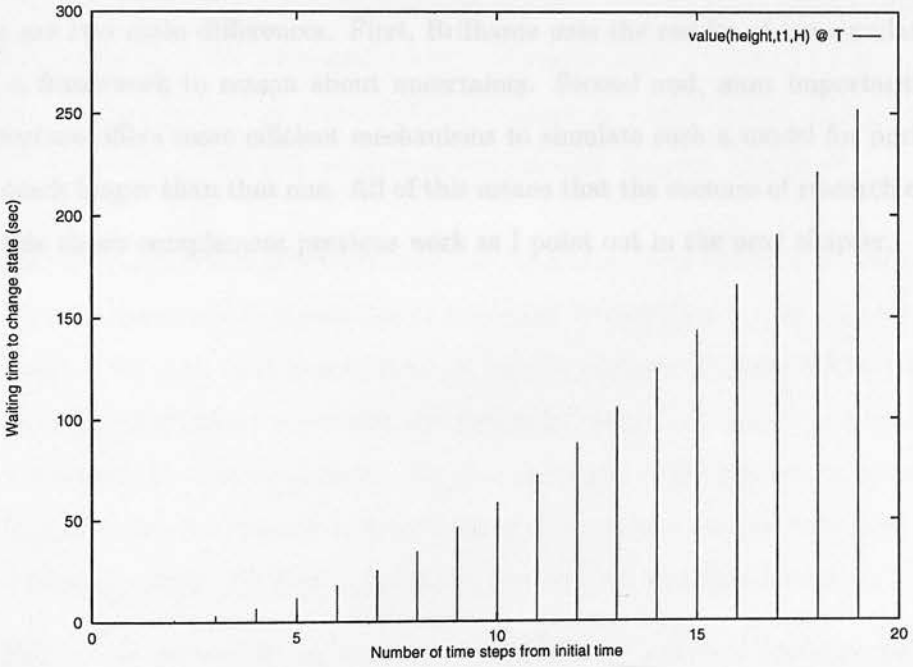

Figure 7.11: Performance of a **NatureTime** interpreter which cache lemmas.

One very important thing to note is that in works like these ones mentioned above, the waiting time to obtain a solution increases at each time step! This does not happen with ECO-ABS architecture as could we see in the results presented in this chapter, but only when the number of agents increase. Even in this case, Eco-ABS is by far

superior than these approaches when we compare the performance of a **NatureTime** interpreter in Figure 7.11 for 3 agents with Eco-ABS performance running with 10 agents, Figure 7.2. This sort of analysis had not been made by previous work, and a gratifying achievement was to rescue the hopes of using a logic based approach as a framework for prototyping of more complex models.

To our knowledge this is the first medium to large scale simulation of the complexity of typical ecological models made with reasonable performance using a logic based approach. A question which naturally comes is: would it be possible to use Eco-ABS approach to model unreasonably complex model of an actual ecosystem? For example, the one modelled in [Brilhante 96]? The answer is yes, we could do it because the referred work is not a continuous model of an ecosystem, and the structure of the model is very similar to the ones suggested by [Robertson *et al.* 91, Mota 94] using **NatureTime**.

There are two main differences. First, Brilhante uses the results of the simulation to build a framework to reason about uncertainty. Second and, most importantly, our architecture offers more efficient mechanisms to simulate such a model for periods of time much longer than that one. All of this means that the avenues of research opened with this thesis complement previous work as I point out in the next chapter.

Chapter 8

Conclusions and Future Work

We were motivated to carry out this research by the prospect of offering a high level specification framework for simulation of ecological models which could be executed. Our major concern was to provide structures of knowledge which allow modellers to describe complex models involving processes working at many scales of time that could be easily integrated. We have used a new time representation theory to cope with the problem of granularity and embodied its concepts in a distributed AI approach to ecological modelling. We may summarise the previous chapters as follows.

Chapter 1: a hypothetical scenario was used to motivate our discussion on why traditional approaches to simulation of ecological models do not offer suitable mechanisms for high level descriptions of models working at many scales (of time, space, organisation) which can also be easily integrated with other systems and modellers, or decision makers. We also presented some similarities between an object-oriented approach and multi-agent systems and why we have used a logic based approach. We then pointed out the research questions raised.

Chapter 2: we presented the reasons why modelling ecological systems challenges temporal representation and reasoning considering the problems raised with the representation of processes interacting at many scales of time. We presented a simulation clause scheme, addressing its advantages in terms of separating temporal and time independent parts of knowledge and in terms of executing specifications. We then suggested that more general simulation clause schema (GSCS) is necessary to express the "adaptability" of a model with respect to

changes in the running environment (i.e. an environment where ecological agents may be moved in and out).

Chapter 3: a brief theoretical discussion was made concerning the problems of representing cycles and different scales of time. Following this, we proposed an alternative to model both aspects in a Linear-Cyclic hierarchy which avoids the major problems the known temporal logic frameworks have found. **NatureTime** logic was presented and its expressive power to represent the problems discussed in Chapter 2 was evaluated.

Chapter 4: we proposed a temporal substitutional framework to separate standard and temporal unification in order to make a clear and consistent treatment for shared temporal variables within the same specification clause. We presented a meta-interpreter to execute normalised **NatureTime** programs. We then extended the language to deal with aligned and non-aligned interacting agents by adopting a policy which takes the last state of an agent for time stamps in between two consecutive time steps. We discussed the limitations of this approach and addressed its symmetry in relation to alternative solutions. We finally argued that the main limitation arises out of the fact that we simulate groups of agents in a single computational model, no matter what approach is used.

Chapter 5: a perspective of agents from ecological modelling point of view was proposed. The theory is based on the concept of *ecoagent*, which is a building block for the development of simulation models of ecosystems in a multi-agent architecture. *Ecoagents* are grouped into three main classes: passive, reactive and active. the first two are related to concepts we usually associate to ecological entities and the third is related to human notions of acting over ecological models. The main features of an *ecoagent* are: a set of potential sources of influences, where it is defined which classes may affect the agent in the case instances of them satisfy certain constraints; local environment which represents actual instances of potential influences specified.

Actions of agents can be related to the execution of general simulation clause schema which represent such actions. Executing specifications is related to operations over structures to model an agent's potential influence and local environ-

ment. An *envagent* is a special *ecoagent* endowed with mechanisms to coordinate or synchronise the actions of a group. It is also important to keep relevant properties of the group. We have shown that despite the fact that the concepts of *Ecoagency* are specific and more directed toward modelling we may obtain similar representations of *web* and *centrum* via an *ecoagent*'s local environment as the one proposed by a formal language for describing ecological systems.

Chapter 6: an architecture which implements the concepts related to *ecoagency* was developed considering the following basic assumptions. Agents can connect to just one medium of communication; such a medium is an ideal one, i.e. every message reaches its destination; agents execute one message at a time and these arrive in a non-deterministic order; an *envagent* synchronises the actions of agents through time token distribution policy; agents may interact with others in order to obtain resources to execute their actions when their turn come. We have given an informal proof that an Ecological Agent-Based Simulation (Eco-ABS) interpreter is free of deadlocks and starvation.

Chapter 7: we showed how to use Eco-ABS to develop ecological models and presented the results of executing small and large models. We showed how the system scales up when the number of agents increases. The results are gratifying in the sense that it rescues the goal of using a logic based approach for prototyping of complex large models. We discussed the expressive power of Eco-ABS for representing knowledge similar to the one involved in the beginning of the thesis. We finally addressed its limitations and suggested some ways to overcome them.

8.1 Contributions

We claim that the main contributions of this work can be summarised as follows.

8.1.1 A Logic Framework for Building Prototypes of Complex Models

The experiments we carried out demonstrates that we have given a considerable contribution to previous efforts of providing logic-based tools to ecological modelling and simulation [Robertson *et al.* 91, Mota 94, Brilhante 96, Mota *et al.* 96]. By using the

Eco-ABS approach one may have various specifications of combining models, with potential to interact, distributed over a computer network. Instances of such specification (agents) can be executed and, during execution, new components may be introduced or components may be retracted from the system.

This means that modellers no longer need to get too involved in computational aspects of the simulation, but rather in the use of the expressive power of logical languages to describe their understanding about how actual ecosystems work in a declarative way. The importance of this has been already addressed in [Robertson *et al.* 91] and we quote

> "It seems that the use of a well understood, soundly based knowledge representation scheme is particularly important as a means of ensuring that our model comprehension systems are, themselves, comprehensible to others."

The logic notation proposed on Table 6.1 plus the domain dependent functions specification schema (Definitions 6.12–6.14) provides a high level representation framework where no knowledge about elementary instructions of a programming language are going to be used to manipulate this knowledge. This provides proper abstractions for modellers to build their models in a more effective way as pointed out in Section 1.3. The underlying mechanisms of execution can be any one, but this thesis provides a *multi-ecoagent* architecture as addressed below in Section 8.1.3.

8.1.2 A Declarative Theory of Time

The separation of the temporal part of the knowledge from other parts of it is helpful to make modellers of ecosystems think first about relations between components of models, and then think when such relations should hold. Such a separation should be supported by a framework of time which has as much structural similarity as possible with temporal phenomena in nature. With such a framework we should be able to easily represent many scales of time and cyclicity of processes. This work proposes one framework which meets these criteria.

By using the declarative theory of time we propose modellers can specify the levels of

time she/he wants to work with; and the separation between relations among objects and when such relations hold through time is represented by using a logic programming notation augmented with special operators and temporal labels. This fits nicely in a general simulation clause representation.

Cyclical processes are elegantly represented by using syntactical representation of temporal cycles. This means that modellers do not have to see the recurrence of such processes by means interpretation of universally quantified temporal variables over (a potentially) infinite set.

Patterns for the combination of models depend on the mechanisms that such models have to reason about processes aligned and non-aligned in time. The framework proposed here is neatly extended to allow this.

8.1.3 A Distributed AI Solution to Integrate Different models

For more complex problems, due to the well known computational limitations of classical AI in relation to the huge increase in the search space, a multi-agent architecture representation is used. As a result we are able to investigate how such patterns of interaction scale from finer to coarser levels of time and structure. We propose a theory (Section 6.5 until Section 6.8) where models are called *ecoagents* and can be combined into groups. The specification of groups can be done in a similar fashion to the specification of its components. This provides a standard way of defining models in a similar way of defining groups (Section 6.3), and so scaling up can be more easily obtained.

The set of potential influences of a class of *ecoagents* suggests that we may have other sorts of interacting agents, provided they all agree in their underlying theory of influence, i.e. what *ecoagents* are and how they relate to each other.

8.2 Discussion and Further Work

It is hard to construct a high level description formalism for ecological simulation without any influence of the features of the domain. To establish clearly where domain features and patterns of representation and inference meet is easily obtained by the

use of schemes such as the ones proposed in Chapter 6. Given that we have seen some of these thresholds one natural question is how far are we from the goal of making the arguments embodied in simulation models accessible to people who must use the models to make decisions [Robertson *et al.* 91] or to other sorts of users as discussed in our motivation in Section 1.1?

Suppose one user (or group of users) have a set F of reactive and passive ecoagents representing a model of an ecosystem, another set S of active *ecoagents* representing a social model about a given community living in F, and another set E of active *ecoagents* representing a economical and legal models about E. Then, it would be ideal if we could test the possible interactions between these models. It is unquestionable the gains we would have by achieving such a long term goal.

The purpose of this work was to provide a more structured formalism to develop models which fall into F category. As *ecoagents* are supposed to run with a certain degree of autonomy and modularity, it is expected that further work to investigate the most convenient ways to represent the interactions with other categories should not affect models developed for E.

From the results obtained in this research there are some interesting avenues which are worth investigating in order to make this goal a little closer than we have reached so far. We summarise of some these roads as follows.

- What sorts of temporal knowledge assessment (or queries) would be interesting from a modeller's point of view, a student point of view, a decision maker point of view?
- Would this need more complex temporal expression involving other temporal entities such as collection intervals?
- To what extent would such queries influence the performance of system applications for large and complex models of ecosystems?
- Would those queries involve qualitative or uncertain knowledge? In this cases, is the representational framework expressive enough? What sorts of structures and inference mechanisms should be added to the architecture to cope with it?

In the case of qualitative reasoning, an interesting avenue of application is modelling [Salles & Bredweg 97] approach to qualitative modelling of the interactions between populations and environmental factors. How this would be related to the potential influences of an ecoagent as proposed here?

- Are there patterns of Eco-ABS specifications that we could gather in a sort of Eco-ABS technique library to be re-used? An interesting investigation is to find out how the patterns of logic programming techniques generation for ecological modelling [Castro 95] suits into our approach.

As far as Eco-ABS paradigm is concerned we identify the following directions for further work.

The potential influence definition scheme needs more expressiveness to be able to capture knowledge about an *envagent* and the attributes of it. Such attributes do not need to be represented in the *envagent* but distributed in the set of agents which composes it.

The *ecoagent* theory needs to consider that assimilation, sensing and acting function should have more general mechanisms related to processing messages. This might be the link with the investigations on languages for agent communication and interaction for negotiation.

An explicit representation of **Net of Potential Influences** is necessary when a considerable number of different classes is expected in the application domain. Although this might be the case for models of ecosystems, the examples investigated in this work took a few classes only to show the fundamental ideas of the theory.

Finally, it would interesting to know how *ecoagency* functional perspective of attributes, processes and functions is related to the usual notion of deliberative agents. To what extend could such alleged correspondence help us to delineate limits of efficient computation of such theories?

Appendix A

NatureTime Semantics

A.1 Linear-Cyclic Hierarchy of Time

This work assumes points as primitives temporal entities. A point will be represented by a set with only one element. An interval is a set of points with the first and last elements as the ending points of the interval. This approach is similar to [Shoham 88a]. For the purposes of this work I assume that points range over $\mathbb{Z}$ (which I shall also write $\mathcal{E}$), but other choices are possible (*e.g.* one may assume it just as a set of states of the world). In the following definitions $2^{\mathcal{E}}$ represents subsets of $\mathcal{E}$; $\mathbb{Z}_{m_i}$ represent the set i of integers modulo m_i (simply modular set).

Definition A.1 (Linear Collapsi Function) *Let* $\mathbb{Z}_{m_1}, \mathbb{Z}_{m_2}, \ldots, \mathbb{Z}_{m_{n-1}}$ *be modular sets,* $\mathbb{Z}$ *the set of positive integers. Then the* Linear Collapsi Function $\mu_\circ$ *is a function which maps* $\mathbb{Z}_{m_1} \times \mathbb{Z}_{m_2} \times \ldots \mathbb{Z}_{m_{n-1}} \times \mathbb{Z}$ *to* $\mathbb{Z}$, *i.e.* $\mu_\circ : \mathbb{Z}_{m_1} \times \mathbb{Z}_{m_2} \times \ldots \mathbb{Z}_{m_{n-1}} \times \mathbb{Z}^n \rightarrow \mathbb{Z}$. *If* $x_1 \in \mathbb{Z}_{m_1}, \ldots, x_n \in \mathbb{Z}$, *then*

$$\begin{aligned}\mu_\circ(x_1, \ldots, x_n) = {} & x_n \times m_{n-1} \ldots \times m_2 \times m_1 + \\ & x_{n-1} \times m_{n-2} \times \ldots m_2 \times m_1 \ldots + \\ & x_2 \times m_1 + x_1,\end{aligned}$$

A sequence of assertions defining elements of the language we call MTCs is actually related to a temporal structure of time defined as follows.

Definition A.2 (Linear-Cyclic Hierarchy of Time) *A* Linear-Cyclic Hierarchy *of*

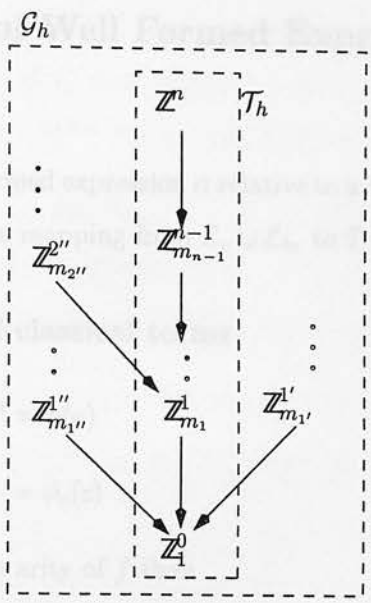

Figure A.1: Induced graph of time granularity. The MTCs inside the innermost dashed box compose $\mathcal{T}_h$.

time $\mathcal{LC}_H$ *is* $\langle \mathcal{G}_h, \ominus \rangle$, *where*

- $\mathcal{G}_h$ *is a tree where the vertices* $\mathcal{V}$ *are modular sets. By adding a distinguished leaf node with label* $\mathbb{Z}$ *we end up with a structure like the one depicted in Figure A.1. Thus, augmented* $\mathcal{G}_h$ *has a subset* $\mathcal{T}_h$ *which defines a unique directed path from the highest set of integers to the root.*
- $\ominus_{m_i}$ *is a partial order relation over the nodes of* $\mathcal{G}_h$.

A **NatureTime** temporal model is a structure $\mathcal{M} = \langle \mathcal{D}, \psi_c, \psi_f, \mathcal{LC}_H, \tau_c, \tau_p \rangle$, where $\mathcal{D}$ is a set of non temporal individuals, ψ_c is a map from $\mathcal{L}_c$ to $\mathcal{D}$, ψ_f is a map from $\mathcal{L}_f$ to $\mathcal{D}^n \to \mathcal{D}$, $\mathcal{LC}_H$ is a Linear-Cyclic hierarchy of time, τ_c is a map from $\mathcal{L}_{tc}$ to $\mathcal{V}$, $\tau_p : \mathcal{L}_p \times \mathcal{D}^n \times \mathcal{E} \to \{\top, \bot\}$. A **NatureTime**-Interpretation is a triple $\mathcal{H}_{nt} = \langle \mathcal{M}, g, h \rangle$ where $\mathcal{M}$ is a temporal model and g is a function which assigns values to the variables of **NatureTime** such that $g : \mathcal{L}_v \to \mathcal{D}$ and $h : \mathcal{L}_{tv} \to 2^{\mathcal{E}}$.

Definition A.3 (Inclusion between two Intervals) *Let* $x_1, y_1, x_2, y_2 \in \mathbb{Z}, x_1 \leq y_1$, $x_2 \leq y_2$, $I_1, I_2 \in 2^{\mathcal{E}}$ *such that* $I_1 = \{z \mid x_1 \leq z \leq y_1\}$ *and* $I_2 = \{z \mid x_2 \leq z \leq y_2\}$. *Then,* I_1 *is included in* I_2 *if* I_1 *is a subset of* I_2, *i.e.* $I_1 \subseteq I_2$.

A.2 Denotation of Well Formed Expressions and Formulae

The denotation of a well-formed expression α relative to a **NatureTime**-Interpretation $\mathcal{H}_{nt}$, written as $[\![\alpha]\!]^{\mathcal{H}_{nt}}$, is a mapping from $\mathcal{L}_v \cup \mathcal{L}_{tv}$ to $\mathcal{D} \cup 2^{\mathcal{E}}$ defined as follows.

A.2.1 Denotation of classical terms

- if $v \in \mathcal{L}_v$, then $[\![v]\!]^{\mathcal{H}_{nt}} = g(v)$
- if $c \in \mathcal{L}_c$, then $[\![c]\!]^{\mathcal{H}_{nt}} = \psi_c(c)$
- if $f \in \mathcal{L}_f$ and n is the arity of f then

  $$[\![f(x_1, \ldots, x_n)]\!]^{\mathcal{H}_{nt}} = \psi_f(f)([\![x_1]\!]^{\mathcal{H}_{nt}}, \ldots, [\![x_n]\!]^{\mathcal{H}_{nt}})$$

A.2.2 Denotation of temporal terms

In what follows, if S is a set then $|S|$ is the number of elements in S.

- if $v \in \mathcal{L}_{tv}$ then $[\![v]\!]^{\mathcal{H}_{nt}} = h(v)$
- if $c \in \mathcal{L}_{tc}$, then $[\![c]\!]^{\mathcal{H}_{nt}} = \tau_c(c)$
- if t is of the form $f(\ldots)$ for $f \in \mathcal{L}_{tf}$ then in the case t is
  - $t(x_1, \ldots, x_n)$, then $[\![t(x_1, \ldots, x_n)]\!]^{\mathcal{H}_{nt}} = \{\mu_\circ(x_1, \ldots, x_n)\}$
  - $t(x_1, \ldots, x_n)...t(y_1, \ldots, y_n)$, then $[\![t(x_1, \ldots, x_n)...t(y_1, \ldots, y_n)\ ]\!]^{\mathcal{H}_{nt}}$ is $\{z \in \mathbb{Z} \mid \mu_\circ(x_1, \ldots, x_n) \leq z \leq \mu_\circ(y_1, \ldots, y_n)\}$.
  - $i(x...y, c_j)$ where $x, y \in \mathbb{Z}_{m_j}$ of $c_j \in \mathcal{L}_{tc}$ and $[\![c_j]\!]^{\mathcal{H}_{nt}} \in \mathcal{T}_h$, then $[\![i(x...y, c_j)]\!]^{\mathcal{H}_{nt}}$ is $\bigcup\{[\![t(u_1, \ldots, u_{j-1}, x, v_{j+1}, \ldots, v_k)...t(v_1, \ldots, v_{j-1}, y, v_{j+1}, \ldots, v_k)]\!]^{\mathcal{H}_{nt}}$ such that $u_1 \in \mathbb{Z}_{m_1} \ldots u_k \in \mathbb{Z}_{m_k}, v_1 \in \mathbb{Z}_{m_1} \ldots v_k \in \mathbb{Z}_{m_k}\}$
  - $i(c_i(x)...c_i(y), c_j)$ where $c_i, c_j \in \mathcal{L}_{tc}$ and $x, y \in \mathbb{Z}_{m_j}$ of c_j and $[\![c_j]\!]^{\mathcal{H}_{nt}} \notin \mathcal{T}_h$, then $[\![i(c_i(x)...c_i(y), c_j)]\!]^{\mathcal{H}_{nt}} = \bigcup_{k=0,1,\ldots}\{z \mid k \times m_j + x \leq z$ and if $x \leq y$ then $z \leq k \times m_j + y$, otherwise $z \leq (k+1) \times m_j + y$ and $mod\_temp\_class(c_j, c_i, m_j)\}$.

– $i(c_i(x_1)...c_i(x_2), c_j)$ *of* $i(y_1...y_2, c_k)$ where $c_i, c_j, c_k \in \mathcal{L}_{tc}$ and $[\![c_i]\!]^{\mathcal{H}_{nt}}$, $[\![c_k]\!]^{\mathcal{H}_{nt}} \in \mathcal{T}_h$ and $[\![c_j]\!]^{\mathcal{H}_{nt}} \notin \mathcal{T}_h$ then $[\![i(c_i(x_1)...c_i(x_2), c_j)\ of\ i(y_1...y_2, c_k)]\!]^{\mathcal{H}_{nt}} =$ is $[\![i(c_i(x_1)...c_i(x_2), c_j)]\!]^{\mathcal{H}_{nt}} \cap [\![i(y_1...y_2, c_k)]\!]^{\mathcal{H}_{nt}}$

- $p(s, c_i)$, $s \in \mathbb{Z}^+, c_i \in \mathcal{L}_{tc}, \mathbb{Z}_{m_i}, \mathbb{Z}_{m_{i-1}}, \ldots, \mathbb{Z}_1$ is path from the level of c_i until the root of $\mathcal{G}_h$, $I \in \mathcal{I}_L$ and it is ground, then

  $[\![p(s, c)]\!]^{\mathcal{H}_{nt}} = |[\![I]\!]^{\mathcal{H}_{nt}}|$ such that $|[\![I]\!]^{\mathcal{H}_{nt}}| = s \times m_i \times m_{i-1} \times \ldots \times 1$.

- if r is a temporal entity, p is a period of time then

  $[\![p\ after\ r]\!]^{\mathcal{H}_{nt}} = [\![r]\!]^{\mathcal{H}_{nt}}\ \omega_{\oplus}\ [\![p]\!]^{\mathcal{H}_{nt}}$, where $\omega_{\oplus}$ is the up-wave modular sum (Appendix B.2).

- if r is a temporal entity, p is a period of time then

  $[\![p\ before\ r]\!]^{\mathcal{H}_{nt}} = [\![r]\!]^{\mathcal{H}_{nt}}\ \omega_{\ominus}\ [\![p]\!]^{\mathcal{H}_{nt}}$, where $\omega_{\ominus}$ is the up-wave modular subtraction (Appendix B.2).

- if $T \in \mathcal{T}_M$ or $T = p\ after\ t$ and $c_1 \in \mathcal{T}_h$, then take $\lfloor T \rfloor$ as shorthand for $p(1, c_1)$ *before* T.

A.2.3 Denotation of Classical and Temporal Formulae

- if $\alpha \in \mathcal{L}_p$ and n is the arity of α, then

  $[\![\alpha(x_1, \ldots, x_n)]\!]^{\mathcal{H}_{nt}} = \top$ iff $\tau_p(\alpha, \langle [\![x_1]\!]^{\mathcal{H}_{nt}}, \ldots, [\![x_n]\!]^{\mathcal{H}_{nt}} \rangle, t) = \top$ for all $t \in \mathcal{E}$. This means that this AF is time-independent, and so does not need to be annotated with a PTE.

- if $\alpha(x_1, \ldots, x_n)$ @ T is an ATF and T a PTE, then

  $[\![\alpha(x_1, \ldots, x_n)\ @\ T]\!]^{\mathcal{H}_{nt}} = \top$ iff $\forall t \in [\![T]\!]^{\mathcal{H}_{nt}}\ \tau_p(\alpha, [\![x_1]\!]^{\mathcal{H}_{nt}}, \ldots, [\![x_n]\!]^{\mathcal{H}_{nt}} \rangle, t) = \top$

- if A and B are ATFs, then

  $[\![A \& B]\!]^{\mathcal{H}_{nt}} = \top$ iff $[\![A]\!]^{\mathcal{H}_{nt}} = \top$ and $[\![B]\!]^{\mathcal{H}_{nt}} = \top$.

  $[\![A \vee B]\!]^{\mathcal{H}_{nt}} = \top$ iff $[\![A]\!]^{\mathcal{H}_{nt}} = \top$ or $[\![B]\!]^{\mathcal{H}_{nt}} = \top$.

Definition A.4 (Logical Consequence) *A formulae B follows from a formula A iff every temporal model $\mathcal{M}$ and variable assignments g and h where A is true also makes B true.*

A.2.4 Auxiliary Procedures

Assuming that the user has given a time hierarchy description as defined in Section 3.5.2, some auxiliary procedures are defined as follows.

- if $r = t(x_1, \ldots, x_k)$ and $r = t(y_1, \ldots, y_k)$, then $precedes(r, s)$ is true if either

  $x_k < y_k$ or

  $x_i < y_i$ and $\forall x_j, y_j (i < i \leq k \vee x_j \leq y_j)$

- if r and s are moments of time, p a period of time then

  $future(r, p, s)$ is true iff $[\![s]\!]^{\mathcal{H}_{nt}} = [\![p\ after\ r]\!]^{\mathcal{H}_{nt}}$

- if $P = p(\Delta, c_i)$, $\Delta \in \mathbb{Z}^+$ and $c_i \in \mathcal{L}_{tc}$ and $P' \in \mathcal{P}$ then
  $down\_equivalent(P, P')$ is true if either

  - $P = P'$
  - $P' = p(\Delta \times m, c_{i-1})$ and $mod\_temp\_class(c_i, c_{i-1}, m)$.
  - $down\_equivalent(p(\Delta \times m, c_{i-1}), P')$ is true and
    $mod\_temp\_class(c_i, c_{i-1}, m)$.

- if $P_1, P_2 \in \mathcal{P}$ then
  $equivalents(P_1, P_2)$ is true iff $\exists P \in \mathcal{P}$ such that $down\_equivalent(P_1, P)$ holds and $down\_equivalent(P_2, P)$ holds.

- if $P, P' \in \mathcal{P}$, $c_i, c_1 \in \mathcal{L}_{tc}$ then
  $mod\_decomposed(P, c_i, P')$ is true iff either

  - P is a single period and $down\_equivalent(P, p(\Delta_1, c_1)$ holds and
    $down\_equivalent(p(1, c_i), p(\Delta_2, c_1)$ holds and
    $\Delta = \Delta_1$ **div** Δ_2, $\Delta' = \Delta_1$ **mod** Δ_2, and $P' = p(\Delta', c_1)$ *plus* $p(\Delta, c_i)$.
  - $P = P_1$ *plus* P_2 and $down\_equivalent(P_1, p(\Delta_1, c_1))$ holds and
    $down\_equivalent(P_2, p(\Delta_2, c_1))$ holds and
    $down\_equivalent(p(1, c_i), p(\Delta_3, c_1))$ holds and for $\Delta = \Delta_1 + \Delta_2$ **div** Δ_3, either

    $P' = p(\Delta, c_i)$ just if $\Delta_1 + \Delta_2$ **mod** $\Delta_3 = 0$ or

$$P' = p(\Delta', c_1)\ \mathit{plus}\ p(\Delta, c_i)\ \text{if}\ \Delta_1 + \Delta_2\ \mathbf{mod}\ \Delta_3 = \Delta'.$$

Appendix B

Some Important Properties

B.1 Modular Inclusion Relation

Properties of $\ominus$ - In what follows, $\mathcal{C}_i$ means the MTC at level i. This relation establishes a sub-division relationship between MTCs. The $\ominus$ relation has the following properties.

- *transitive* - if $\mathcal{C}_i$, $\mathcal{C}_j$ and $\mathcal{C}_k$ are MTCs and $\mathcal{C}_k \ominus \mathcal{C}_j$ and $\mathcal{C}_j \ominus \mathcal{C}_i$, then $\mathcal{C}_k \ominus \mathcal{C}_i$.
- *reflexive* - if $\mathcal{C}_i$ is a MTC then $\mathcal{C}_i \ominus \mathcal{C}_i$ (every MTC can be subdivided by itself)
- *anti-symmetric* - if $\mathcal{C}_i$, $\mathcal{C}_j$ are MTCs, and $i \neq j$, and $\mathcal{C}_j \ominus \mathcal{C}_i$, then $\mathcal{C}_i \ominus \mathcal{C}_j$.

B.2 Up-Wave Modular Sum and Subtraction

Definition B.2.1 *Let* $P = p(\Delta, c_i)$, $mod\_temp\_class(c_{i+1}, c_i, m)$, *and* $t(s_1, \ldots, s_k)$. *The* up-wave modular sum *between* P *and* I, $I\,\omega_{\oplus}\,P$, *is defined as*

$$I\omega_{\oplus}P = \begin{cases} t(s_1, \ldots, s_i + \Delta, \ldots, s_k) & \text{if } s_i + \Delta < m, \\ t(s_1, \ldots, s_i \oplus \Delta, \ldots, s_k)\omega_{\oplus}P' & \text{if } s_i + \Delta \geq m, \text{ and } P' = p(\Delta', c_{i+1}), \\ & \text{where } \Delta' = s_i + \Delta \text{ div } m. \end{cases}$$

Definition B.2.2 *Let* $P = p(\Delta, c_i)$, *and* c_i *defines another MTC as* $mod\_temp\_class(c_{i+1}, c_i, m)$, *and* $I = t(s_1, \ldots, s_k)$. *We call the* up-wave modular subtraction *between* P *and* I, $I\,\omega_{\ominus}\,P$, *defined as*

$I\,\omega_{\ominus}\,P = t(s_1, \ldots, s_i - \Delta, \ldots, s_k)$, *iff* $\Delta < s_i$

$I\ \omega_{\ominus}\ P = t(s_1, \ldots, s_i \ominus \Delta, \ldots, s_k)\ \omega_{\ominus}\ P'$, *iff* $\Delta >= s_i$ *and* $P' = p(\Delta', c_{i+1})$, *where* $\Delta' = \Delta\ div\ m$.

B.3 Temporal Unification

B.3.1 Specification of $temp\_unify/3$

In this section I shall define the unification of temporal terms or time matching. It consists basically on reducing pure temporal expressions and performing a temporal reasoning to check whether or not two temporal terms have the same interpretation wrt a MoTT.

$temp\_unify(T, T, UT) \Longleftarrow reduce(Term, UTerm).$

$temp\_unify(T_1, T_2, UT) \Longleftarrow$

$\neg\ T_1 = T_2\ \&$

$reduce(T_1, RT_1)\ \&$

$reduce(T_2, RT_2)\ \&$

$reduce(UT, RUT)\ \&$

$unify\_units(RT_1, RT_2, RUT).$

The reduction of temporal terms is given as follows.

B.3.2 Reduction of Temporal Terms

The the reduction algorithm is divided into two groups of clauses. The first consists of the clauses to deal with canonical forms of temporal expression. The second deals with complex forms. First we introduce what we mean by temporal variable: a term t is a temporal variable if it is a logical variable, i.e. if $t \in \mathcal{L}_v$, or a canonical form of PTE where all its elements are logical variables. Based on it we have the following reduction algorithm.

Definition B.3.1 *Let t_1 be a PTE, and t_2 a canonical PTE. We say that t_1 is reducible to t_2, written $reduce(t_1, t_2)$ iff one of the following holds.*

- $t_1 = t_2$, *and* t_1 *is a temporal variable,* $temp\_var(t_1)$.
- $\neg\ temp\_var(t_1)$ *and* t_1 *is in the form* $t(x_1, \ldots, x_k)$, *so* $t_2 = t_1$
- $t_1 = s_1 ... s_2$, *and* $t_2 = rs_1 ... rs_2$ *where*

  s_1 *is reducible to* rs_1 *and*

  s_2 *is reducible to* rs_2.
- $t_1 = t_2 = i(s...t, c)$
- $t_1 = \Delta\ after\ t$ *and*

  t_1 *is reducible to* rt_1 *and*

  t_2 *is reducible to* rt_2 *and*

  $future(rt_1, \Delta, rt_2)$ *holds.*

B.3.3 Unification of Time Units

In what follows we present the cases for temporal unification of time units or temporal entities. This level can be considered as the core of the temporal reasoning process of the whole temporal unification. The simple matching for all cases is $unify\_units(T, T, T)$. The others are given as follows, where $functor(Term, F, N)$ is true if the function name of $Term$ is F and $Term$ has arity N, $time\_instace(I, C, LI)$ is true if LI is one linear time interval instance [1] of the cyclical interval I of modular class C, $level(C, N)$ is true if the level of modular temporal class C is N.

- Matching of Linear Intervals

  $unify\_units(T_1 ... T_2, T_3 ... T_4, TR) \Longleftarrow linear\_unify(T_1, T_2, T_3, T_4, TR)$.

  where $linear\_unify/5$ implements the well know matching relations of inclusion, overlapping, etc.[Allen & Hayes 85].

- Matching cyclical or modular intervals

[1] This is according to the principle of linear realizability a circular time structure [Rescher & Urquhart 71]

$unify\_units(i(I_1, C), i(I_2, C), i(IR, C)) \Longleftarrow$

$mod\_temp\_class(\_, C, M)$ &

$integer(M)$ &

$modular\_unify(I_1, I_2, IR)$.

where $modular\_unify/3$ implements the cyclical version of the matching relations proposed by Allen and Hayes (*op. cit.*) and described in Section B.3.4.

- Matching cyclical intervals to get one linear instance of it

$unify\_units(i(I_1, C), i(I_2, C), TR) \Longleftarrow$

$\neg var(TR)$ &

$functor(TR, ..., 2)$ &

$mod\_temp\_class(\_, C, M)$ &

$integer(M)$ &

$modular\_unify(I_1, I_2, IR)$ &

$time\_instance(IR, C, TR)$.

$unify\_units(i(I, C), T_1...T_2, TR) \Longleftarrow$

$time\_instance(I, C, T_3...T_4)$ &

$linear\_unify(T_1, T_2, T_3, T_4, TR)$.

$unify\_units(T_1...T_2, i(I, C), TR) \Longleftarrow$

$time\_instance(I, C, T_3...T_4)$ &

$linear\_unify(T_1, T_2, T_3, T_4, TR)$.

- Matching a moment of time with a cyclical interval

$unify\_units(T, i(I, C), T) \Longleftarrow$

$functor(T, t, \_)$ &

$mod\_temp\_class(flow\_time, Class, infinite)$ &

$level(Class, N)$ &

$arg(N, T, Arg)$ &

$time\_instance(I, C, T_1...T_2)$ &

$unify\_moment\_interval(T, N, Arg, T_1...T_2)$.

$unify\_units(i(I, C), T, T) \Longleftarrow unify\_units(T, i(I, C), T)$

B.3.4 Matching Relations Between Cyclical Intervals

In this work, matching between cyclical intervals is made considering they are at the same level of time hierarchy. The possible cases are EMPTY INCLUDED and OVERLAPPING. In the following analysis of cases we are considering two things. First, that the ordering of the circular set is like the clock, and that $s_1...t_1$ is the range of interval I_1 and $s_2...t_2$ is the range of interval I_2. Second, we will assume linearity between the elements, by picking the first without the circular link with the last.

1. EMPTY match - there are two different cases with the empty result, or one fail result, as shown in figure B.1. For each case we have the following conditions, and the result of the match for all of them is the same, that is the empty set.

   (a) $t_2 > s_2$ and $s_2 > t_1$ and $t_1 > s_1$

   (b) $s_2 > t_1$ and $t_1 > s_1$ and $s_1 > t_2$

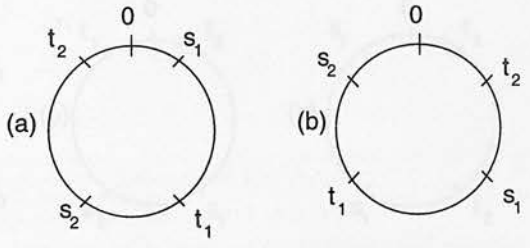

Figure B.1: Empty matching of modular intervals.

2. INCLUDED Match - there are four different cases of one interval is included in th other as showed in figure B.2. For each case we have the following conditions, and the result of the match for all of them is the same, that is $s_2...t_2$.

   (a) $t_1 \geq t_2$ and $t_2 > s_2$ and $s_2 \geq s_1$

   (b) $t_2 > s_2$ and $s_2 \geq s_1$ and $s_1 > t_1$

   (c) $s_1 > t_1$ and $t_1 \geq t_2$ and $t_2 > s_2$

   (d) $s_2 \geq s_1$ and $s_1 > t_1$ and $t_1 \geq t_2$

3. OVERLAPPING Match - there are 6 different cases of overlap between two modular intervals. They are divided in two groups

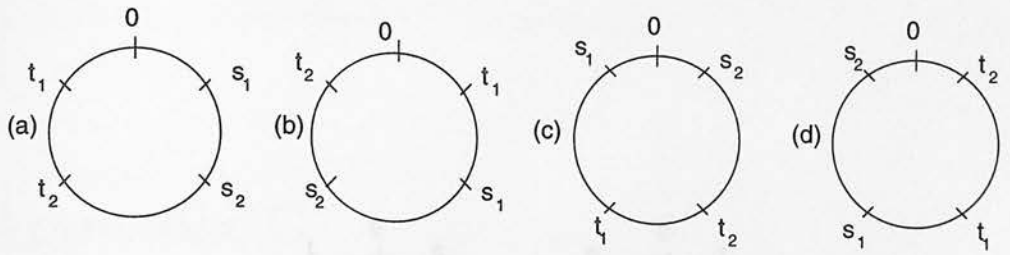

Figure B.2: Matching of one interval including other.

i the cases showed in figure B.3. For each case we have the following conditions, and the result of the match for all of them is the same, that is $s_2...t_1$

(a) $t_2 > t_1$ and $t_1 \geq s_2$ and $s_2 > s_1$

(b) $t_1 \geq s_2$ and $s_2 \geq s_1$ and $s_1 > t_2$

(c) $s_2 > s_1$ and $s_1 > t_2$ and $t_2 > t_1$

(d) $s_1 > t_2$ and $t_2 \geq t_1$ and $t_1 \geq s_2$

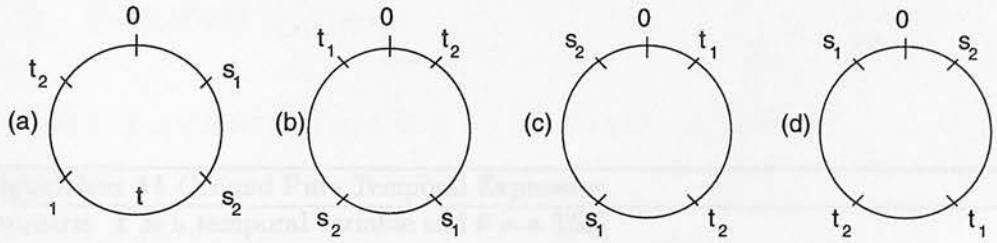

Figure B.3: First Four cases of overlapping between intervals.

ii the cases shown in figure B.4. For each case we have the following conditions, and the result are two possible intervals $s_1...t_2$ and $s_2...t_1$

(a) $t_2 \geq s_1$ and $s_1 > t_1$ and $t_1 \geq s_2$

(b) $s_2 > t_2$ and $t_2 \geq s_1$ and $s_1 > t_1$

In this case, when matching the intervals we expect to obtain only one answer, and the other can be computable as an alternative matching. Computationally speaking it is normally done by backtracking.

B.3.5 Ground Pure Temporal Expression

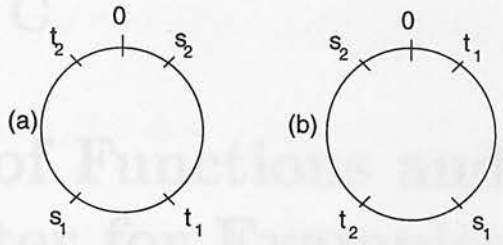

Figure B.4: The Last two cases of overlapping between intervals.

Algorithm 11 Ground Pure Temporal Expression

Require: T is a temporal variable and θ is a TSF.

Ensure: Either T is *t-bound* to a ground pure temporal expression V in relation to θ, written $ground\_pte(T, \theta, T, V)$, or it returns false otherwise.

Case 1: $ground\_pte(T, \theta, V)$ if
$var(T)$
V is $t\_bind(T, \theta)$
$\neg\ var(V)$.

Case 2: $ground\_pte(P\ after\ T, \theta, P\ after\ V)$ if
$var(T)$
$temp\_subst(T, \theta, V)$.

Case 3: $ground\_pte(P\ after\ T, \theta, P\ after\ T_1)$ if
$\neg\ var(T)$
$\neg\ ground(T)$
$ground\_pte(T, \theta, T_1)$.

Appendix C

Library of Functions and Interpreter for Examples

In this appendix I present some examples of function definitions used bye the simulations showed in Chapter 7.

C.1 Function Schema

$function(logistic, par(H, Hmax, Gr), H * (1 + Gr * (1 - H/Hmax))).$

$function(block_movement, par(H), H).$

$function(shadow, par(H, D), -(H/(D * 1000))).$

$function(peste, par(Pos), -(Pos/10000)).$

$function(same\_value, par(V), V).$

$function(average, par(L), average(L)).$

$function(sum, par(L), sum(L)).$

C.2 Library of Execution Functions

$execute(logistic, Att, V_i, Infs, V) \Longleftarrow$

$max(Att, MAX)$ &

$change\_rate(Att, CR)$ &

$RealCR\ is\ CR + Infs$ &

$function(logistic, par(Vi, MAX, RealCR), Expr)$ &

$eval(Expr, V)$.

$execute(up\_zig\_zag, Altitude, Ap, Infs, Af) \Longleftarrow$

$max(Altitude, Max)$ &

$((Ap > 2 * Infs/3$ & $Af\ is\ Ap - (0.8 + 3 * Ap/(5 * Max)))$

$\vee$

$(Ap =< 2 * Infs/3$ & $Af\ is\ Ap + 1 + Ap/Max))$.

$execute(flat\_zig\_zag, Pos, Pp, Infs, Pf) \Longleftarrow$

$max(movement_x, Max_X)$ &

$random\_value(0.0, Max\_X, D_x)$ &

$max(movement_y, Max_Y)$ &

$random\_value(0.0, Max_Y, D_y)$ &

$RD_x\ is\ D_x + Infs$ &

$RD_y\ is\ D_y + Infs$ &

$translade(Pp, d(RDx, RDy, 0.0), Pf)$.

Appendix D

Glossary

D.1 Mathematical and (Temporal) Logical Symbols

- $\mathbb{N}$- The set of natural numbers
- $\mathbb{Z}_m$ - Set modulo m or modular set with modular value m
- $\neg$ - Negation by failure
- $\Longleftarrow$ - Logical implication (reverse notation)
- $\&$ - Logical conjunction
- $\vee$ - Logical disjunction
- $\top$ - Truth value true
- $\bot$ - Truth value false
- $\mathcal{L}_v$ - Set of symbols for variables
- $\mathcal{L}_c$ - Set of symbols for constants
- $\mathcal{L}_f$ - Classical functions
- $\mathcal{L}_p$ - Predicates
- $\mathcal{L}_{tv}$ - Set of symbols for temporal variables
- $\mathcal{L}_{tc}$ - Finite set of temporal constants

- $\mathcal{T}_C$ - Set of names of temporal classes
- $\mathcal{L}_{tf}$ - Set of temporal function symbols
- $\mathcal{L}_p$ - Finite set of predicate symbols
- $\mathcal{T}_M$ - Set of temporal terms which represent moments of time.
- $\mathcal{I}_L$ - Set of temporal terms which represent linear intervals.
- $\mathcal{I}_o$ - Set of temporal terms which represent cyclical intervals.
- $\mathcal{P}_{TE}$ - Set of pure temporal expressions.
- @ - Temporal symbol to annotate a classical formula
- $\lfloor\ \rfloor$ - Temporal interval demarcator
- $\mathcal{M}$ - It is a **NatureTime** temporal model.
- $\mho$ - An *ecoagent* architecture.
- $\mathcal{N}_{eco}$ - The set of all names of an *ecoagent* architecture.
- $\mathcal{N}_{cls}$ - The set of all names of classes of *ecoagents*, and $\mathcal{N}_{cls} \subset \mathcal{N}_{eco}$.
- $\mathcal{N}_{ag}$ - The set of all names of *ecoagents*, and $\mathcal{N}_{ag} \subset \mathcal{N}_{eco}$.
- $\mathcal{N}_{Att}$ - The set of all name of attributes of classes of *ecoagents*, and $\mathcal{N}_{Att} \subset \mathcal{N}_{eco}$.
- $\mathcal{N}_{ler}$ - The set of all names of local environmental relations, and $\mathcal{N}_{ler} \subset \mathcal{N}_{eco}$.
- Σ_Λ - The set of all possible sets of attributes (or states).
- $\mathcal{P}_{inf}$ - The set of potential influences.
- $\mathcal{F}_{inf}$ - The set of functions of influence upon agents' processes.
- $\Sigma_{A_{res}}$ - The set of resource assimilation functions related to resource assimilation factor.
- $\mathcal{I}_{act}$ - The set of internal actions or computational functions associated to actions.
- Θ - An *ecoagent*'s local environment.

- Σ_Θ - The set of all possible local environments.
- $\mathcal{S}_\Theta$ - It is a function for sensing the environment, used to generate the agent's initial local environment.
- $\mathcal{SC}_\Theta$ - It is a function for sensing changes in the agent's local environment.
- $\mathcal{S}_\Gamma$ - It is a function for sensing information resource.
- $\mathcal{M}_\mho$ - The tuple which represents a Multi-Ecoagent architecture.
- $\textcircled{\mho}$ - The special agent which represents properties of the environment or the *envagent.*
- $\Sigma_\mho$ - A set of of *ecoagents.*
- $\mathcal{K}_{ext}$ - The "artificial" *Environmental Medium* to represent external knowledge and interaction among agents.
- Ω - The *medium of communication.*
- $\overrightarrow{out}$ - The function function which maps the set of messages an agent may send Ω to Ω.
- $\overleftarrow{in}$ - The function which maps the set of messages an agent may receive and Ω to Ω.
- Λ - The state of an *ecoagent.*
- Γ - An *ecoagent* information resource structure.
- $\mathcal{Q}$ - The list of messages an agent has to attend.
- $\mathcal{G}$ - The group structure representation of an *envagent.*
- $\wp$ - A list which represents an *envagent* time stamp scheduler of actions.
- Σ_v - The set of terms of the form $value(Att_j, agent(C))$ or $absorbed(amount(X, Att_j), agent(C))$.
- Σ_h - The set of terms of the form $holds(loc\_env\_rel(R, A, agent(Cls)))$ or $holds(loc\_env\_rel(R, agent(Cls), A))$.
- Σ_{inf} - The set of terms of the form $func(\pi, n, k)$.

D.2 Special Constants, Terms, Temporal Functions and Predicates

- *lowest* - Special constant used to limit the lowest level c_1 of a hierarchy of modular sets.
- *flow_time* - Special constant used to limit the highest level c_n of a hierarchy of modular sets.
- *infinite* - Special constant used to say that the flow of time is an infinite sequence of events of the highest level of time.
- *smallest* - Special constant used to say that the lowest level of granularity is the smallest interval or a moment in time.
- .../2 - Special constant used to represent the range between two positive numbers.
- $p/2$ - Special constant used to represent the length of time (first argument) at a given scale (second argument).
- $i/2$ - Special constant used to represent a cyclical interval (first argument) at a given level of time (second argument).
- t/k - Special constant used to represent a moment of time in a time hierarchy in which the MTH has k elements.
- $of/2$ - Special constant used to compose collection intervals.
- $before/2$ - Special constant used to represent a displacement function which computes a moment in time in the past given a current moment and the length of time between them.
- $after/2$ - Special constant used to represent a displacement function which computes a moment in time in the future given a current moment and the length of time between them.
- *plus*/2 - Special constant used to compose lengths of time at different scales.

- *mod_temp_class*/3 - Special predicate used to declare a temporal class (first argument) as being a modular set of another one (second argument) with modular value defined in the third argument.

- *future*/3 - Special predicate to relate a temporal entity in time, a period of time and a temporal entity in the future.

- *precedes*/2 - Special predicate to represent a relation of precedence between two temporal entities.

- *down_equivalent*/2 - Special predicate which associates a period of time, at a given scale, with an equivalent period at a lower scale.

- *equivalents*/2 - Special predicate used to compare if two periods of time are equivalent to each other.

- *mod_decomposed*/3 - Special predicate used to associate a period of time and an scale of time to a decomposition of the given period at the given scale.

- *solve*/3 - Special predicate used to define the meta-interpreter or proof procedure for **NatureTime** logic.

- *body_of_simu_clause*/1 - Special predicate used to classify formulas which have instances in the body of a simulation clause.

- *clause*/1 - Special predicate used by the meta-interpreter for testing the existence of a clause in its temporal normal form.

- *inst_simucls*/1 - Special predicate used to classify clauses which have a form of a simulation clause.

- *free_search*/5 - Special predicate which implements a forward chaining search used by the meta-interpreter.

- *sub_tsf*/3

- *ground_pte*/1 - Special predicate used to classify temporal terms which are ground instances of pure temporal expressions.

- *temp_comb*/5 - Special predicate used to associate two temporal variables and their respective TSF to a TSF which is the result of a temporal combination between such *t-variables*.
- *temp_subst*/3
- *aligned_search*/6
- *change*/2 - Special predicate used to relate an agent's attribute and the process which changes the attribute through the flow of time.
- *scale*/2 - Special predicate used to relate a process and the scale of time at which the process works.
- *smaller_period*/2 - Special predicate used to test whether or not a period of time (the first argument) is smaller than another one (the second argument).
- *search*/5 - Special predicate called by the meta-interpreter to derive a sequence of models which satisfy a simulation clause in which the head is a fixed moment of time.
- *length_of_time*/2 - Special predicate used to associate a time interval and the length of such period including both ending points of the interval.
- *compute_rest*/7 - Special predicate called by the meta-interpreter to compute the rest of the progress of the value of an attribute after having computed its first value.
- *progression*/6 - Special predicate called by the meta-interpreter to which implements the observer process of an agent.
- *progress*/3 - Special predicate of the **NatureTime** language which wraps the observer process of ana agent.
- *ecoagent*/6 - Special predicate used as the external interface of an *ecoagent*.
- *attribute*/4 - Special predicate used to define an agent's attribute.
- *initial_value*/2 - Special predicate used to define the initial value of an attribute.

- *max*/2 - Special predicate used to define the maximum value of an agent's attribute.
- *compute*/2 - Special predicate to relate a process and the name of the function used to compute the action of the process.
- *affect*/5 - Special predicate used as a meta-predicate to define potential influences over agents.
- *send*/1 - Special predicate used by *ecoagent* meta-interpreter to send a message to other agents.
- *broadcast*/1 - Special predicate used by *ecoagent* meta-interpreter to broadcast a message to other agents.
- *multicast*/2 - Special predicate used by *ecoagent* meta-interpreter to multi-cast a messages to other agents.
- *write_ek*/1 - Special predicate used by *ecoagent* meta-interpreter to write the external interface of an agent on the *environmental medium.*
- *read*/1 - Special predicate used by *ecoagent* meta-interpreter to read (and extract) a message from the *medium of communication.* If no message is present, then a call to this predicate fails.
- *wait*/1 - Special predicate used by *ecoagent* meta-interpreter to read (and extract) a message from the *medium of communication.* If no message is present, then the agent execution is blocked until a message is available.
- *wait_rd*/1 - Special predicate used by *ecoagent* meta-interpreter to just read a message from the *medium of communication.* If no message is present, then the agent execution is blocked until a message is available.
- ϵ - A special term which represent an *ecoagent* state of computation.
- *may_influence*/3 - A special predicate which associates the attributes of an agent A to its set of potential influences.
- *classes_inf*/2 - A special predicate which associates an agent's set of potential influences to a set of names of classes of influences within it.

- *location*/2 - A special predicate which associates an agent's initial state or goal to its intended location within the environment.

- *field_influence*/2 - A special predicate which associates the set of potential influences of agent to its maximum field of influence.

- *actions*/4 - Special predicate which associates an agent's initial time and its attributes to the time limit of observation of its sensing messages and a list of scheduled actions according to their time scale and (possibly) other constraints.

- *sense_inf*/7 - Special predicate which associates an agent, the next set of actions to be executed, the agent's local environment, its information resource structure, its set of potential influences and the interval of sensing messages to the initial information resource structure and the number of messages expected to fulfill it.

- *lc*/7 - Special predicate called by the agent's meta-interpreter which starts an agent's life cycle.

- *envagent*/5 - Special predicate used as the external interface of an *envagent*.

- $\Im$ - A special term which represent an *envagent* state of computation.

- *active_envagent*/1 - Special predicate which call the meta-interpreter for the execution of an *envagent*'s behaviour.

- *starting_time*/1 - Special predicate which takes the initial time of a multi-ecoagent simulation.

- $\mathcal{I}_{cond}$/2 - Special meta-predicate to represent the set up of an *envagent*'s initial state (geographical position, area, etc.).

- *env_lc*/2 - Special predicate which activates the life cycle of an *envagent*.

- *process*/4 - Associates a message, an *envagent* and its current state of computation to a new state of computation after processing the given message.

- *zones_influenced*/3 - Special predicate which associates an agent's location and its ratio of influence to the zones within *envagent*'s are which have intersection with its ratio.

- *insert*/3 - Special predicate which associate an element and a set to another set equal to the first but added with the element.
- *agents_inf*/4 - Special predicate which associates a set of zones in the *envagent* area, the whole group of agents and a set of classes of influence to a set of agents from this class which are currently present in the zones.
- *insert_into_group*/5 - Special predicate which associates an agent, its location, its class and a group structure to a new group added with this agent.
- *apply_policy*/3 - Special predicate which associates an *envagent*, its current state of computation to a new state of computation after applying the *time-token* distribution policy.
- *insert_req*/5 - Special predicate which associates an *envagent*, and *ecoagent*, the agent's time stamp request and the scheduler of actions of the *envagent* to a new scheduler of actions.
- *already_moved*/3 - Special predicates which test whether an agent of a certain class has been moved from a given *envagent* or not.
- *dispatch*/4 - Special predicate which associates an *envagent*, a term with an answer received from the environment and the group information and the list of queries made to a new list of queries after having processed the dispatch(or not) of the query associated to the message received.
- *execute*/5 - Special predicate used by the *ecoagent* interpreter to call a domain dependent function which changes the value of an attribute, taking the previous value and the influences over the attribute into account.
- *function*/3 - Special predicate to define a function by its name, its interface (a sequence of parameter where no typing theory is assumed), and and expression or body which relates the parameters.

D.3 Terms and Expressions

- Temporal Substitutional Framework (TSF) - a set of pairs where each pair represents a temporal variable and its binding.
- Temporal Normal Form (TNF) - a temporal formula written in a way in which its temporal terms and their bindings are all gathered in the TSF of the formula.
- *t-free* - A temporal variable in a TSF in which its binding is a non-instantiated variable.
- *t-bound* - a temporal variable in a TSF in which its binding is instantiated to a temporal term.
- *Semantic Unification* - the unification between two function terms which takes a special theory about them into account.
- *Temporal Unification* - a semantic unification which interprets specially labelled terms according to a specific structure of time. This process matches them by performing temporal reasoning which considers their meaning in the given time representation.
- **NatureTime**-Resolution - A special resolution rule which is composed of two parts: *i*) standard unification step involving non temporal terms, and *ii*) a temporal combination involving temporal terms and their bindings in the TSFs of both formulas involved in the resolution step.
- *Ecoagent* - Is a short name for Ecological Agent which is a building block for the development of simulation models of ecosystems in a multi-agent architecture.
- Potential Influences - a set of meta-relations about the potential sources of influence among agents from different classes.
- Local Environment - is a dynamic structure containing the instantiation of potential influences affecting an agent.
- *Envagent* - Special *ecoagent* endowed with mechanisms for coordinating a group of agents, and keeps general information about the whole environment.

- *time-token* - A time stamp requested by an agent and offered by *envagent*.

- Time Token Distribution Policy - Is an expression to mean that agents which have the preference to act are those which (also) attend the policy of having the earliest time stamp for request of execution.

Bibliography

[Abel & Niven 90] D. E. Abel and B. S. Niven. Application of a formal specification language to animal ecology. i. environment. *Ecological Modelling*, 50(1):205–212, 1990.

[Agha 86] Gul Agha. *Actors - A Model for Concurrent Computation for Distributed Systems.* The MIT Press, 1986.

[Allen & Hayes 85] James F. Allen and P. J. Hayes. A commonsense theory of time. In *Proceedings of the of the 9th IJCAI*, pages 528–531, 1985.

[Allen 83] James F. Allen. Maintaining knowledge about temporal intervals. *Communications of the ACM*, 26(11), 1983.

[Austin 62] J.L. Austin. *How to Do Things With Words.* Oxford University Press, 1962.

[Barbuceanu & Fox 95] Mihai Barbuceanu and Mark S. Fox. Cool: A language for describing coordination in multi-agent systems. In *First International Conference on Multi-Agent Systems*, pages 17–24, San Francisco, California, 1995.

[Barringer *et al.* 89] Howard Barringer, Michael Fisher, Dov Gabbay, and Anthony Hunter. Metatem: A framework for programming in temporal logic. In *REX Workshop on Stepwise Refinement of Distributed Systems: Models, Formalisms, Correctness.* LNCS Volume 430, Springer Verlag, 1989.

[Baveco & Lingeman 92] J. M. Baveco and R. Lingeman. An object-oriented tool for individual-oriented simulation: host-parasitoid system application. *Ecological Modelling*, 61(1):267–286, 1992.

[Baveco & Smeulders 94] J.M. Baveco and A.M.W. Smeulders. Objects for simulation: Smalltalk and ecology. *Simulation*, 61(1):42–57, 1994.

[Biggs 87] Norman L. Biggs. *Discrete Mathematics.* Oxford Science Publication, 1987.

[Bouzid & Ladkin 95] Maroua Bouzid and Peter Ladkin. Rules for simple temporal reasoning. In *2nd International Workshop on Temporal Representation and Reasoning, Florida USA*, 1995.

[Brazier *et al.* 95] Francez Brazier, Barbara Dunin Keplicz, Nick R. Jennings, and Jan Treur. Formal specification of multi-agent systems: a real-world case. In *1st International Conference on Multi-Agent Systems*, pages 25–32, San Francisco, California, 1995.

[Brilhante 96] Virginia V. B. Brilhante. INFORM-logic: a system for representing uncertainty in ecological models. Unpublished M.Sc. thesis, Department of Artificial Intelligence/University of Edinburgh, 1996.

[Burkhart 94] Roger Burkhart. The Swarm multi-agent simulation system. In *OOPSLA*, Sept. 1994.

[Capra 92] Fritjof Capra. *The Tao of Physics.* Flamingo, Third Edition - 1992.

[Castro 95] Alberto Castro. A techniques-based framework for program generation in ecological modelling. DAI Discussion Paper 159, Department of Artificial Intelligence, University of Edinburgh, April 1995.

[Ciapessoni *et al.* 93] E. Ciapessoni, E. Corsetti, A. Montanari, and P. San Pietro. Embedding time granularity in a logical specification language for synchronous real-time systems. *Science of Computer Programming*, 20(1):141–171, 1993.

[Cohen & Levesque 90] P. R. Cohen and H. J. Levesque. Intention is choice with commitment. *Artificial Intelligence*, 42(1):213–261, 1990.

[Cohen & Levesque 95] Philip R. Cohen and Hector J. Levesque. Communicative actions for artificial agents. In *1st International Conference on Multi-Agent Systems*, pages 65–72, San Francisco, California, 1995.

[Cukierman & Delgrand 95] Diana Cukierman and James Delgrand. A language to express time intervals and repetition. In *Proceedings of the of 2nd International Workshop on Temporal Representation and Reasoning*, Melbourne Beach, Florida - USA, April 1995.

[Davies 95] Paul Davies. *About Time.* Penguin Books, 1995.

[Davis & Smith 83] Randall Davis and Reid G. Smith. Negotiation as a metaphor for distributed problem solving. *Artificial Intelligence*, 20(1):63–109, 1983.

[Dean 89] Thomas Dean. Using temporal hierarchies to efficiently maintain large temporal databases. *Journal of the ACM*, 36(4):687–718, 1989.

[Einstein 23] Albert Einstein. *The Principle of Relativity.* Dover Publications - New York, 1923.

[Engelfriet & Treur 94] Joeri Engelfriet and Jan Treur. Temporal theories of reasoning. In C. MacNish, D. Pearce, and L. M. Pereira, editors, *European Workshop JELIA - Logics in AI*, Lecture Notes in Artificial Intelligence, pages 279–295. Springer-Verlag, 1994.

[Fagin *et al.* 95] Ronald Fagin, Joe Halpern, Yoram Moses, and Moshe Vardi. *Reasoning about knowledge.* MIT Press, 1995.

[Ferber & Jacopin 91] Jacques Ferber and Eric Jacopin. The framework of eco-problem solving. In Yves Demazeu and Jean-Pierre Muller, editors, *DECENTRALIZED A. I. - 2*, pages 181–193. Elsevier Science Publishers, 1991.

[Ferber 96] Jacques Ferber. Reactive distributed artificial intelligence: Principles and applications. In G.M.P. O'Hare and N.R. Jennings, editors, *Foundations of Distributed Artificial intelligence*, Sixth General Computer Technology, pages 287–314. John Wiley & Sons, Inc, 1996.

[Ferreira 95] J. G. Ferreira. Ecowin - an object-oriented ecological model for aquatic ecosystems. *Ecological Modelling*, 79(1):21–34, 1995.

[Feyman 92] Richard P. Feyman. *The Character of Physical Law.* Penguin Books, 1992.

[Fiadeiro & Maibaum 94] Jose Luiz Fiadeiro and Tom Maibaum. SOMETIMES "TOMORROW" IS SOMETIME action refinement in a temporal logic of objects. In *First International Conference on Temporal Logic*, pages 48–66, Bonn, Germany, July 1994. Springer-Verlag.

[Fisher & Barringer 91] M. Fisher and Howard Barringer. Concurrent MetateM - a language for distributed AI. In *European Simulation Multiconference*, Copenhagen, Denmark, June 1991.

[Fisher & Keane 95] Michael Fisher and John Keane. Realising a concurrent object-based programming model on parallel virtual shared memory architectures. In *Massively Parallel Programming Models*, October 1995.

[Fisher & Wooldridge 94] M. Fisher and M. Wooldridge. Specifying and executing protocols for cooperative action. In *International Working Conference on Cooperating Knowledge-Based Systems (CKBS)*, Keele, U.K., June 1994.

[Fisher 93] M. Fisher. Concurrent MetateM - a language for modeling reactive systems. In *Parallel Architectures and Languages Europe*, Munich,Germany, June 1993.

[Fisher 94a] M. Fisher. A survey of concurrent MetateM - the language and its applications. In *First International Conference on Temporal Logic(ICTL)*, Bonn, Germany, July 1994.

[Fisher 94b] Michael Fisher. Representing and executing agent-based systems. In *ECAI Workshop on Agent Theories,Architectures and Languages (ATAL)*, Amsterdam, Netherlands, August 1994.

[Fisher 96] Michael Fisher. Temporal semantics for concurrent MetateM. *Journal of Symbolic Computation*, 22(5–6):627–648, 1996.

[Folse *et al.* 89] L. J. Folse, M. Packard J, and W. E. Grant. AI modelling of animal movements in a heterogeneous habitat. *Ecological Modelling*, 46(1):57–72, 1989.

[Frisch & Page 95] Alan M. Frisch and C. David Page. Building theories into instantiations. In *Proceedings of the Fourteenth IJCAI, August*, pages 1210–1216, 1995.

[Frisch 91] Alan M. Frisch. The substitutional framework for sorted deduction: fundamental results on hybrid reasoning. *Artificial Intelligence*, 1(49), 1991.

[Frolund 96] Svend Frolund. *Coordinating Distributed Objects.* The MIT Press, 1996.

[Gabbay 87] Dov Gabbay. *Modal and Temporal Logic Programming. In Temporal Logics and their Applications - Antony Galton Editor.* ACADEMIC PRESS, 1987.

[Gabbay 89] Dov Gabbay. The declarative past and imperative future. In B. Bonieqbal and H. Barringer, editors, *Coloquium on Temporal Logics and Specification*, pages 409–448. Springer Verlag, 1989. V(398).

[Gavrila & Treur 94] Ioa Gavrila and Jan Treur. A formal model for the dynamics of compositional reasoning systems. In A. Cohn, editor, *11th ECAI*, pages 307–311, Amsterdam, Holland, 1994. John Wiley & Sons, Ltd.

[Goodwin 92] Richard Goodwin. Formalizing properties of agents. Technical Report CMU-CS-93-159, Carnegie Mellon University - Department of Computer Science, Pittsburg, USA, 1992.

[Gotelli 95] Nicholas J. Gotelli. *A Primer of Ecology.* Sinauer Associates, Inc. - Publishers, 1995.

[Haddadi 95] Afsaneh Haddadi. Communication and cooperation in agent systems. In G. Goss, J.Hartmanis, and J. van

Leeuwen, editors, *Lecture Notes in Artificial intelligence.* Springer Verlag, 1995.

[Hewitt 77] Carl Hewitt. Viewing control structures as patterns of passing messages. *Artificial Intelligence*, 8(1):323–364, 1977.

[Hewitt 88] Carl Hewitt. Offices are open systems. In Alan H. Bond and Les Gasser, editors, *Readings in Distributed Artificial intelligence*, pages 321–329. Morgan Kaufmann, Inc, 1988.

[Hiebeler 94] David Hiebeler. The Swarm simulation system and individual-based modelling. In *Decision Support 2001: Advanced Technology for Natural Resource Management*, Toronto, Canada, Sept. 1994.

[Hobbs 85] Jerry R. Hobbs. Granularity. In *Proceedings of the 9th IJCAI*, pages 1–4, 1985.

[Hogger 84] Christopher John Hogger. *Introduction to Logic Programming.* Academic Press, Inc., 1984.

[Kardec 57] Allan Kardec. *Le Livre des Espirits.* Éditions DERVY, 1857.

[Kardec 68] Allan Kardec. *La genèse.* Éditions DERVY - Paris, 1868.

[Kreifelts & vonMartial 91] Thomas Kreifelts and Frank von Martial. A negotiation framework for autonomous agents. In Yves Demazeu and Jean-Pierre Muller, editors, *DECENTRALIZED A. I. - 2*, pages 71–88. Elsevier Science Publishers, 1991.

[Ladkin 86a] Peter Ladkin. Primitives units for time specification. In *Proceedings of the AAAI*, pages 354–359, 1986.

[Ladkin 86b] Peter Ladkin. Time representation: A taxonomy of interval relations. In *Proceedings of the AAAI*, pages 360–366, 1986.

[Ladkin 87] Peter Ladkin. The completeness of a natural system for reasoning with time intervals. In *Proceedings of the AAAI*, pages 462–467, 1987.

[Leban *et al.* 86] Bruce Leban, David D. McDonald, and David R. Foster. A representation for collections of temporal intervals. In *Proceedings of the AAAI*, pages 367–371, 1986.

[Levin 92] Simon A. Levin. The problem of pattern and scale in ecology. *Ecology*, 73(6):1943–1967, 1992.

[Liu & Orgun 96] Chuchang Liu and Mehmet A. Orgun. Dealing with multiple granularity of time in temporal logic programming. *Journal of Symbolic Computation*, 22(5–6), 1996.

[Lloyd 84] John W. Lloyd. *Foundations of Logic Programming.* Springer Verlag, 1984.

[Luck & d'Inverno 95] Michael Luck and Mark d'Inverno. A formal framework for agency and autonomy. In *1st International Conference on Multi-Agent Systems*, pages 254–260, San Francisco, California, 1995.

[Makela *et al.* 93] M. E. Makela, G. A. Rowell, W. J. Sames IV, and L. T. Wilson. An object-oriented intracolonial and population level model of honey bees based on behaviours of European and Africanized subspecies. *Ecological Modelling*, 67(1):259–284, 1993.

[McCarthy & Hayes 81] John McCarthy and Patrick Hayes. Some philosophical problems from the standpoint of artificial intelligence. In *Readings in Artificial intelligence*, pages 431–450. Tioga Publishing Co., 1981.

[McDermot 82] Drew McDermot. A temporal logic for reasoning about processes and plans. *Cognitive Sciences*, 6(2), 1982.

[Minsky 86] Marvin Minsky. *The Society of Mind.* Basic Books, 1986.

[Mix *et al.* 92] Michael C. Mix, Paul Farber, and Keith I. King. *Biology - The Network of Life.* Harper Collins Publishers, 1992.

[Montanari 94] Angelo Montanari. A metric and layered temporal logic for time granularity, synchrony and asynchrony. First International Conference on Temporal Logic, July 1994.

[Moore 95] Robert C. Moore. *Logic and Representation.* CSLI Publications, 1995.

[Mota & Robertson 96] Edjard Mota and David Robertson. Representing interaction of agents at different time granularities. In *3rd International Workshop on Temporal Representation and Reasoning, Key West - Florida USA.* DAI RP-796, Edinburgh University, 1996.

[Mota 94] Edjard Mota. Temporal representation of ecological domains. DAI TP- 31, Department of Artificial Intelligence, University of Edinburgh, 1994.

[Mota *et al.* 95] Edjard Mota, Mandy Haggith, Alan Smaill, and David Robertson. Time granularity in simulation models of ecological systems. In *Workshop on Excutable Temporal Logics- Montreal, Canada.* DAI RP-740, Edinburgh University, 1995.

[Mota *et al.* 96] Edjard Mota, David Robertson, and Alan Smaill. NatureTime: Temporal granularity in simulation of ecosystems. *Journal of Symbolic Computation*, 22(5–6):665–698, 1996.

[Muller 96] H. Jurgen Muller. Negotiation principles. In G.M.P. O'Hare and N.R. Jennings, editors, *Foundations of Distributed Artificial intelligence*, Sixth General Computer Technology, pages 211–230. John Wiley & Sons, Inc, 1996.

[Newton-Smith 80] W. H. Newton-Smith. *The Structure of Time.* Routledge & Kegan Paul Ltd, 1980.

[Niven & Abel 91] B. S. Niven and D. E. Abel. Logical synthesis of environment of king penguin, aptenodytes patagonicus. *Ecological Modelling*, 56(1):291–311, 1991.

[Niven 82] B. S. Niven. Formalization of the basic concepts of animal ecology. *Erkenntnis*, 17(1):307–320, 1982.

[Niven 87] B. S. Niven. Logical synthesis of an animal's environment: Sponges to non-human primates. *Australian Journal of Zoology*, 35(1):597–606, 1987.

[Olson & Sequeira 95] Richard L. Olson and Ronaldo A. Sequeira. An emergent computational approach to the study of ecosystem dynamics. *Ecological Modelling*, 79(1):95–120, 1995.

[Orgun & Wadge 92] Mehmet A. Orgun and W. W. Wadge. Theory and practice of temporal logic programming. In L. Fari nas del Cerro and M. Penttonen, editors, *Intensional Logics for Programming*, pages 23–50. Oxford University Press, 1992.

[Pachet *et al.* 95] Francois Pachet, Geber Ramalho, Jean Carrive, and Guillaume Cornic. Representing temporal musical objects and reasoning in the muses system. In *International Congress on Music and Artificial Intelligence*, pages 33–48, 1995.

[Poidevin 95] Robin Le Poidevin. Relationism and temporal topology: Physics and metaphysics? In Robin Le Poidevin and Murray MacBeath, editors, *The Philosophy of Time*, Oxford Readings in Philosophy, pages 149–167. Oxford University Press, 1995.

[Rescher & Urquhart 71] Nicholas Rescher and Alasdair Urquhart. *Temporal Logic.* Springer Verlag, 1971.

[Robertson *et al.* 91] David Robertson, Alan Bundy, Robert Muetzelfeldt, Mandy Haggith, and Michael Uschold. *Eco-Logic Logic-Based Approaches to Ecological Modelling.* The MIT Press, 1991.

[Rosenschein & Zlotkin 94] Jeffrey S. Rosenschein and Gilad Zlotkin. Desining conventions for automated negotiation. *Artificial Intelligence Magazine*, pages 29–46, 1994. Fall.

[Salles & Bredweg 97] Paulo Salles and Bert Bredweg. Building qualitative models in ecology. In *11th International Qualitative Reasoning Workshop, Italy, June 3–6*, 1997.

[Shoemaker 95] Sydney Shoemaker. Time without change. In Robin Le Poidevin and Murray MacBeath, editors, *The Philosophy of Time*, Oxford Readings in Philosophy, pages 63–79. Oxford University Press, 1995.

[Shoham 88a] Yoav Shoham. *Reasoning about Change – Time and causation from Standpoint of Artificial Intellgence.* The MIT Press, 1988.

[Shoham 88b] Yoav Shoham. Temporal logics in AI: Semantical and ontological considerations. *Artificial Intelligence*, 33(1), 1988.

[Shoham 93] Yoav Shoham. Agent-oriented programming. *Artificial Intelligence*, 60(1):51–92, 1993.

[Sichman & Demazeau 92] Jaime Simao Sichman and Yves Demazeau. When can knowledge-based systems be called agents? In *IX Brazilian Symposyum on Artificial Intelligence (SBIA)*, Rio de Janeiro, Brazil, 1992.

[Singh 94] Munidar P. Singh. *Multiagent Systems - A Theorretical Framework for Intentions, Know-How, and Commitments.* Number 799 in Lecture Notes in Artificial Intelligence. Springer-Verlag, 1994.

[Sterling & Shapiro 86] Leon Sterling and Ehud Shapiro. *The Art of Prolog.* The MIT Press, 1986.

[van Emden 84] M. H. van Emden. An interpreting algorithm for prolog programs. In *Prolog Implementatins*, Ellis horwood Series ARTIFICIAL INTELLIGENCE, pages 93–110. Ellis Horwood Ltd, 1984.

[Wooldridge & Jennings 95] Michael Wooldridge and Nicholas R. Jennings. Intelligent agents: theory and practice. *The Knowledge Engineering Review*, 10(2):115–152, 1995.

Published Papers

During the PhD course, papers and article describing the research in this thesis were presented in workshops and published in journals as presented below.

1. Edjard Mota, Mandy Haggith, Alan Smaill and David Robertson - Time Granularity in Simulation Models of Ecological Systems . In Proceedings of the Workshop on Executable Temporal Logics at the IJCAI-95, Montreal - Canada, August 1995.

2. Edjard Mota and David Robertson - Representing Interaction of Agents at Different Time Granularities. Proceedings of the 3rd International Workshop on Temporal Representation and Reasoning - TIME'96, Key West, Florida - USA May 1996. IEEE Computer Society.

3. Edjard Mota, David Robertson and Alan Smaill - NatureTime: Temporal Granularity in Simulation of Ecosystems . Journal of Symbolic Computation - Special Issue on Executable Temporal Logics, V(22), Numbers 5 and 6, December 1996.

Copies of these papers are annexed to this thesis.

Time Granularity in Simulation Models of Ecological Systems

Edjard Mota* ,Mandy Haggith, Alan Smaill, Dave Robertson

The University of Edinburgh,
Department of Artificial Intelligence,
80 South Bridge, Edinburgh EH1 - 1HN, Scotland,
Phone: 44 + 131 + 650- 27{27,07,10,06},
Fax : 44 + 131 + 650-6516,
email: {edjardm,hag,smaill,dr}@aisb.ed.ac.uk

Abstract. Granularity of time is an important issue for the understanding of how actions performed at coarse levels of time interact with others, working at finer levels. However, it has not received much attention from most AI work on temporal logic. In traditional domains of application (e.g. databases, planning, natural language, etc.), we may not need to consider it as a problem, but it becomes important in more complex domains, such as ecological modelling. In this domain, aggregation of processes working at different time granularities is very difficult to do reliably. We have proposed a new time granularity theory based on *modular temporal classes*, and have developed a temporal reasoning system to reason about seasonal cycles. This time theory may be a suitable framework for an executable temporal logic for the specification of ecological models, where each ecological entity is an active agent.

1 Introduction

Temporal reasoning systems usually treat time as a linear sequence of discrete points or linear intervals, and the representation of different levels of time granularity is done by grouping intervals in what are called *convex intervals* [Allen 83], [Allen & Hayes 85], [Ladkin 86], or *non-convex intervals* [Leban *et al.* 86]. In such ontologies of time, cyclical processes are normally represented by using recurrent intervals as in [Koomen 89].

Although many of these theories are formally well developed, very few of them have been used for the specification of simulation models of problems we normally find in nature. The main reason may be the fact that the temporal concepts which people normally use in such domains do not seem to have a direct relation with those of temporal logics. Many phenomena in nature cannot be easily understood in only one scale of time, and using different scales is essential to their comprehension.

* On leave from University of Amazonas, Manaus, Brazil and sponsored by the Coordenation for the Improvement of Academic Staff, CAPES, of the Brazilian Ministry of Education, grant no 01723/93-8.

Nowadays, integration of ecological models is an important issue. There are many individual models of parts of systems, and people want to solve problems which require the behaviour of a number of different models to be combined. However, there is little standardisation in the combination process. This is critical when the models were conceived for different levels of time granularity to simulate processes working at different levels of abstraction, but which are somehow related. Meanwhile, some temporal logics for the specification of reactive systems do not deal with time granularity, and so there is a gap between these areas which we aim to bridge. This cross-fertilisation might result in a new way of modelling ecological systems, and raise other types of temporal problems.

This paper shows a temporal logic which deals with time granularity based on an ontology of time called *Linear-Cyclic Hierarchy*. The logic was successfully used in the representation of seasonal cycles in agroforestry problems, and can be used as a programming language to develop simulation models working at different levels of time granularities, particularly for eco-systems. We also address some points about our current research. This concerns the development of an executable temporal logic for the specification of different simulation models of eco-systems to be integrated. We argue that our hierarchy of time is a suitable base for this logic, where each ecological species is treated as an agent (in the sense of multi-agent systems), each with its own scale of time. In this way, the behaviour of these systems may emerge from the interaction between their component agents.

The rest of this paper is organised as follows. In Section 2, we present the motivation for the research by showing an example of a eco-system model, highlighting the problems raised when simulating such domains. We also present a brief review of temporal logics applied to the ecological domain. In Section 3, we present our theory of time granularity. In Section 4, we show how it can be used to specify simulation models in ecology, particularly in the example shown in Sect. 2. We also address its drawbacks and why its direct use does not lead us to feasible solutions for simulating the interaction of many ecological species. In Section 5, we explain how our time theory could be used in a multi-agent systems architecture to provide an executable temporal logic to attack the problem of integrating different simulation models working at different time granularities. Finally, in Section 6, we present some concluding remarks about the current state of our research and its future.

2 Motivation

The objective of this section is to show the complexity of modelling a real eco-system, the agents involved, and the effect of their actions which can be observed and assumed at various time granularities. At the end of the section, we briefly describe related work and why we did not use traditional temporal logics.

2.1 The Biomass Increase of a Forest

Example 1. A piece of forest, from now on f_1, composed of 10 trees, each one allocated in a square with sides of 3 meters, where the shape of the tree is assumed to be unimportant. Each tree has a growth rate r_i, per some unit of time, which varies according to the season and the level of nutrients in the soil. The growing process of one tree may affect the growing of its neighbours because of the competition for nutrients and light, assuming the absorption of light per area of the canopy of the tree. To give an intuitive idea, the situation is depicted in Figure 1. Each t_i, $i = 1, \ldots, 10$ represents a tree.

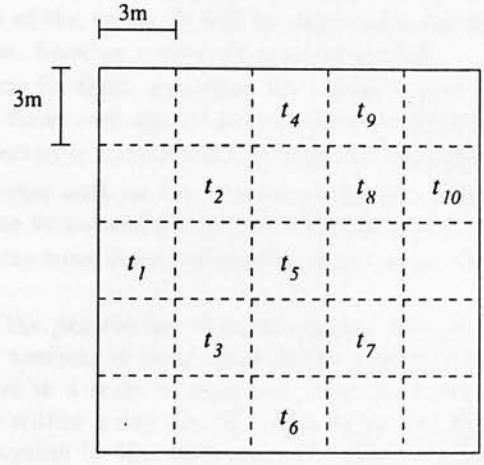

Fig. 1. A simple forest.

The problem of interest in this scenario is to predict the change of height of the trees in a weekly scale of time, and as a consequence their biomass and the biomass of the whole forest. Due the great complexity of the processes involved, what is usually done is to develop a single simulation model, for each process, at an appropriate level of abstraction and time granularity, and after simulating all of them separately the results are joined. But as mentioned in Sect. 1 this practice is very unlikely to give accurate results. For example, the height increase of a tree t_i might be represented by an equation of the form

$$\frac{dH_i}{dt} = r_i H_i (1 - \frac{H_i}{MAX_{H_i}}) \ . \tag{1}$$

where MAX_{H_i} is the maximum height that a tree t_i can reach during its whole life, and r_i is the growth rate of t_i, and H_i is the height of t_i. Usually, r_i is assumed to be an average value per some unit of time. It is intended to represent

all necessary "information" about the growth of t_i. However, this does not model explicitly the interaction among trees, which may affect this rate. So, a more sophisticated simulation model for this problem should consider as a relevant part of the scenario the following processes:

- the rate of water uptake, based on the features of the soil where the tree is placed. This rate would be at some scale of time. Note that the water in-flow of the soil would also change according to the season of the year. So we may need to represent it as a cyclical process.
- the rate of the absorption of nutrients (or uptake), e.g. carbon (photosynthesis) and nitrogen (roots). Another time scale may be needed for this.
- the influence of the concentration of water, nutrients, etc., in the soil, according to the age of the forest. It will be responsible for stopping the increase of tree biomass. Another time scale may be needed.
- the competition for light, assuming the absorption of light per area of the canopy of the trees, and also its height. Another scale may be needed.
- the action of external events which change the environment
  - natural events such as fire, storms, epidemics of insects, new trees appearing due to natural production of seeds, etc.
  - cutting some trees down, reforesting some areas, etc.

Let's consider the process involving the leaves and the roots of a tree, and make an intuitive analysis of their effect in the growth rate. The effect of photosynthesis changes in a scale of minutes, since the incidence of light changes minute by minute within a day. On the other hand, the effects of water uptake and nitrogen absorption by the roots are more effectively described on a hourly time scale. The processes clearly work at different time granularities and we need to integrate them to properly represent their influence in the growth rate of a tree.

For the purpose of this paper, we will concentrate only on the influence of one tree on the growing process of another tree. The influence on a tree t_i by other trees is approximated as a function of their height and their distance from t_i. This is intended to represent how the acquisition of biomass of other trees affects t_i. Basically, the taller a tree t_j, the more effect it will have on the growth rate of t_i, and the further t_j is from t_i the less effect it will cause. This will be represented by the quantity $k\frac{H_j}{d_{ij}}$, where k is a constant which would normally be determined empirically, and d_{ij} is the distance between t_i and t_j. Equation (1) now becomes

$$\frac{dH_i}{dt} = (\sum_{j=1}^{nb} k\frac{H_j}{d_{ij}})H_i(1 - \frac{H_i}{MAX_{H_i}}) \quad . \tag{2}$$

where nb is the number of tree neighbours of t_i.

2.2 Related Work

A direct attempt to model temporal aspects of ecological phenomena in terms of temporal logic, was proposed in [Gray 93], using a linear point structure representation of time. Such an approach does not have any mechanism of inference to reason about simple temporal knowledge, and also nothing about how to solve temporal conflicts on seasonal cycles.

The same problems of reasoning about cyclic events were found when we tried to use other well known logics, such as Gabbay's temporal logic [Gabbay 87]: the temporal logic with *Since* and *Until* and fixed point operators [Gabbay 89], [Barringer *et al.* 91]. Furthermore, these approaches have no commitment to both the "durational" aspects of temporal logic and with different levels of time granularity, but with the specification of possible states of the world in a single time scale.

Another possibility would be Allen's system [Allen 84], [Allen & Hayes 85], where the problem of reasoning about cycles could be implemented using the work on reasoning about recurrences [Koomen 89]. However, these approaches do not clearly explain how to use, in practice, the inference mechanisms they provide to model cyclical events in a "natural" way to obtain appropriate inferences. These logics treat time as a mathematical structure based on natural numbers, or integers, or reals, etc.Almost no attempt has been made to develop a temporal logic theory based on notions of time which we can actually conceive in the natural world, such as day, lunar month, and seasonal cycles, all together. Any temporal logic application, based on whatever mathematical structure, must choose which scale is being regarded. Since we have established that the integration of temporal scales must be an interactive process, it is essential that the notation used to do this can be communicated to the users.

Granularity is very important if we intend to look at the world at different levels of abstraction, and when switching from one level to another it is necessary for the comprehension of the phenomena being observed [Hobbs 85]. This has been applied in the context of temporal databases, in order to obtain efficient maintenance of them [Dean 89], and planning systems, to achieve plan actions at different scales of time and reduce the computational complexity of such systems [Badaloni & Berati 94]. In reactive systems, few temporal logics deal with granularity of time. Recently, [Montanari 94] proposes a many-sorted first order logic augmented with temporal operators and a metric on time to deal with time granularity.

The theory we developed in [Mota 94] was an attempt to provide a logic based language to represent concepts of time, following closely the forms of expression used informally in descriptions of ecological systems in a target domain. In the next section we will present the main rationale of this hierarchical theory of time.

3 NatureTime Logic

In this section we will present our hierarchical theory of time in a very informal way, showing some formalisms only when it becomes necessary. A more formal description can be found in [Mota 94]. To give a better intuition of how such theory of time can be implemented, we will show a simple temporal reasoner interpreter we called **NatureTime**. This is a standard Prolog meta-interpreter with restricted forms of unification on temporal labels.

3.1 Basic Assumptions

Although our logic for reasoning about time and change through time is declarative, it has a procedural (or executable) interpretation because the rules are in the form *future* $\Leftarrow$ *past*. Furthermore, the language used to implement the meta-interpreter was Prolog, which has been the basis for some executable temporal logic definitions due to its procedural interpretation.

In terms of time ontology, instead of simply introspecting on what might be technically possible, and then trying to find examples of such temporal constructs, we did the opposite starting with some domain examples in the form of English text. For instance, about the example showed in Sect. 2.1, one might say "the tree grows faster during the rainy season (say February) than in any other season". By analysing simple statements like this, we first detected the *sorts* of expressions we normally use to refer about inherently cyclical temporal classes, and that are hierarchically related. Afterwards, we attempted to fit them within a many-sorted logic formalism.

This was done by using a different mathematical framework to model temporal classes like "seasons", or cyclical processes. The framework of time we suggested is to model such a *hierarchy of cycles* using modular arithmetic, which is also called clock arithmetic [Biggs 87]. In this view, each cyclical temporal class is defined by one modular set, and so we permit the succession of time in a cyclical way, where the last element is followed by the first. The theory is based on the following assumptions.

1. *temporal entity* (TE) - is a reference to a measure of time, e.g. June, 24th, 1980.
2. TEs are grouped and circularly ordered, forming *modular temporal classes* (MTC), e.g. December and January are TEs of the class "month", and the last element, December, is followed by the first, January.
3. Temporal classes are modularly sub-divided in other temporal classes by what we call *modular value*. For instance, 60 is the modular value which sub-divides hour into minutes.
4. The specification of one modular temporal class defines one level of a time hierarchy. There may exist as many levels of hierarchy as we want. For instance, one could be interested just in days and hours, and so there would be two levels where the second (day) would be defined by the first (hour).
5. The number of levels of a time hierarchy must be finite. The highest can be considered as the larger interval, and the lowest as the smallest.
6. The highest level of the hierarchy is not circularly grouped to form a modular temporal class, but it is linearly ordered in an infinite sequence. Thus, for each instance of the highest temporal class there is one positive integer. In the example of the previous item, day would be a temporal class with its instances linearly ordered, but not those from minutes which would be circularly ordered.

The first allows us to introduce the idea of "previous" and "next" time without inconsistencies, since they will refer to temporal entities at the same level of time granularity. For instance, "next month" would refer to another month, as "previous year" refers to another year.

The second provides a short and elegant mechanism to refer to intermittent intervals. In a *pure* linear structure of time they would be represented by collections of intervals.

The third provides us with a facility to divide a temporal class in the way we need. For instance, if one has defined week as a sequence of seven days, then we can create

labour-week as another modular temporal class. How these two classes would related to each other and other elements of the hierarchy is beyond the scope of this paper.

The fourth allows us to have multiply nested levels of time granularity. For instance, from the example 1 we could define levels of time from years down to minutes, or from months down to hours.

The fifth makes the time ontology a tractable one, since without this restriction there would be no way to compute the operations over our time expressions. This analysis is beyond the scope of the present paper.

Finally the sixth allows us to have the flow of time. If no level were a linear sequence, then this time framework would loop for ever.

3.2 Logic Definition

In what follows, we will use `this font` to refer to the elements of our language. Our logic is an extension of Prolog, and so its syntax is assumed to be same, but enhanced with some special symbols, terms, and a unification algorithm for temporal labels. We are going to show the extensions introduced at the same time as we present our hierarchy of time theory. Afterwards we will show how the logic is implemented.

The vocabulary is formed by the same symbols for variables, constants, functions and predicates as in Prolog. The set of propositional connectives for conjunction, disjunction, and implication are represented here by `&`, `or`, and $\Longleftarrow$, respectively. A classical formula can be "annotated" with temporal expressions, and in this case it is called an *atomic temporal formula* if the associated classical formula is atomic. In the case of a classical well formed formula, when each of its elements or the whole formula is annotated, then we call it a *well formed temporal formula* (WFTF). When a classical formula is not annotated, then it is assumed to mean that it is true in all states of the world.

Temporal expressions, which we also call *pure temporal expressions*, represent the temporal entities of the hierarchy of time which is composed by *modular temporal classes*, or simply MTC. MTCs are defined by using the special predicate

`mod_temp_class(C1,C2,ModularValue)`,

where this relation says that `C1` is modularly defined by `ModularValue` elements of the class `C2`. The hierarchy of time we can define is always a finite one in relation to the number of levels. More formally, a *Linear-Cyclic Hierarchy* of time is a finite ordered set $\mathcal{T}_h$ of modular temporal classes $\{mc_1, \ldots, mc_n\}$, called the set of *time levels*, where

- mc_n = `mod_temp_class(flow_time,`c_n`, infinity)`, where `flow_time` and `infinity` are special constants.
- mc_i = `mod_temp_class(`c_i, c_{i-1}, m_v`)`, $1 < i < n$, where c_i is a constant symbol called the name of the class mc_i, and c_{i-1} is another constant symbol, $c_{i-1} \neq$ `lowest`, which is a special constant, and m_v is a natural number.
- mc_1 = `mod_temp_class(`c_1`, lowest, smallest)`, and c_1 is one constant symbol, and `smallest` is another special constant.

Example 2. A simple version of the calendar model of time can be defined as follows

```
mod_temp_class(flow_time,year,infinity).
mod_temp_class(year,month,12).
mod_temp_class(month,day,30).
mod_temp_class(day,lowest,smallest).
```

This sequence of relation defines the flow of time, represented by `flow_of_time`, as a linear and infinity (`infinity`) sequence of `year`, `year` as a MTC of 12 `month`, `month` as a MTC of 30 `day`[2], `day` as the smallest (`smallest`) interval. Within this hierarchy we have the following recursive definition of sorts of PTEs:

- `t(x1,...,xn)` to represent one single smallest interval of time through the whole hierarchy. For instance, in the hierarchy defined above, the number of levels is n=3, and so `t` is a function with 3 arguments.
- `i(Range,Class)` to represent *cyclical* intervals. For example, `i(11...2,month)` which represents every instance of the interval from November up to February, or the class of all intervals between the November of one year to the February of the next year.
- `P after T` to relate a temporal entity `T` and another temporal entity in the future, in which they are separated by a length of time `P`. `P` is a period of time if it is in the form `p(Value,Class)`, or `p(Value,Class) plus P`, where `plus` is a constructor of composed lengths of time, and `P` is a period. For instance, `p(15,day) after i(2...2,month)` represent "fifteen days after February", and `p(10,day) plus p(3,month) after t(23,3,1994)` represent "3 days and 3 months after March 23, 1994".

The temporal operator we use is the throughout "`@`" operator, which means that a classical logical formula is true throughout the whole interval. It can be defined in terms of the at operator[Fruhwirth 94], for instants of time, or intervals with length zero. In this work we prefer to treat such "instants" of time as small intervals that can always be sub-divided, and so the "`@`" is quite enough to refer to them.

3.3 The Enhanced Meta-interpreter for Temporal Reasoning

In this section we will show the specification of the simple meta-interpreter of the **NatureTime** system. We will show the clauses of the extended temporal meta-interpreter, with an explanation for each extension.

A Solver to NatureTime The meta-interpreter, denominated by *temp_solver*, is an extension of the one presented by [Sterling & Shapiro 86]. The extensions are the introduction of the logical connective to represent conjunction and disjunction; as we have temporal Horn clauses, we will change the predicate `clause` for a convenient representation for temporal Horn clauses; and finally, the introduction of a unification predicate to unify the PTEs. Because of the need to control the unification over PTEs, the standard `solve/1` predicate is changed to be a binary relation `solve/2`. The first argument is the temporal formula to be solved, and the second the result of the computation for the first argument be true. The enhanced meta-interpreter is shown below.

```
solve(A, A):-                          % A is true if
   \+ temp_formula(A),                 % A is not a temporal formula
   A.                                  % and is true as a Prolog goal.
solve(A @ T1, A @ T3):-                % A@T1 is solved to A@T3 if
```

[2] This was done for the sake of simplicity. An extension of this logic is being developed to deal with irregularities such as the real calendar.

```
   \+ A = (_ & _),                    % A is not a composite event
   A @ T2,                            % there is some A@T2 and
   temp_unify(T1, T2, T3).            % T3 is the MGU of T1 and T2.
solve(A @ T,A @ T):-                  % A@T is true if
  X = (A1 & A2),                      % A consists of A1 & A2 events
  solve(A1 @ _,A1 @ T1),              % A1 is true throughout T1
  solve(A2 @ _,A2 @ T2),              % A2 is true throughout T2
  temp_unify(T1,T2,T).                % T is the MGU of T1 and T2.
solve(A & B, A1 & B1):-               % A & B is solved to A1 & B1
   solve(A, A1),                      % if A is solved to A1, and
   solve(B, B1).                      %    B is solved to B1
solve(A or B, R):-                    % A or B is solved to R
   (solve(A, R) ;                     % if A is solved to R, or
    solve(B, R)).                     % if B is solved to R
solve(A @ T1, A @ T3):-               % A@T1 is solved to A@T3 if
   A @ T2 <== Precondition,           % A@T2 is the head of a THC
   solve(Precondition, _),            % The body of THC is true
   temp_unify(T1, T2, T3).            % T3 is the MGU of T1 and T2.
```

The first clause simply verifies if the formula is not a WFTF, and solves the goal by calling a standard Prolog goal. The second clause consults the Knowledge Base (KB) to check whether we can directly deduce `A @ T1` or not. The third clause is to deal with composite events, where we want to know if there is some time interval throughout which they can happen all together. Clauses 4 and 5 are just another way to re-write the standard clauses for logical conjunction and disjunction, and so do not need any detailed explanation. The last clause is the clause for logical implication, which gives an alternative for the first two clauses if they fail. To understand their semantics we need to understand the declarative interpretation of each one; they should be interpreted as follows.

2 - `A` *is true throughout* `T3`, *if* `A` *is not a composite event, and* `A` *is true throughout* `T2`, *and it is true that* `T3` *is the temporal unification of* `T1` *and* `T2`.

3 - `A` *is true throughout* `T` *if, A is a composite event consisting of* `A1 & A2`, *and A1 is true throughout* `T1`, *and A2 is true throughout* `T2`, *and* `T` *is the temporal unification of* `T1` *and* `T2`.

6 - `A` *is true throughout* `T3` *if, the* `A @ T2` *is the head of a temporal Horn clause with body* `Precondition`, *and the body can be solved, and* `T3` *is the temporal unification of* `T1` *and* `T2`.

Unification Algorithm The specialised unification algorithm we developed to treat PTEs consists of two steps. First, every complex PTE is reduced to a canonical form of temporal unit. Second, canonical forms of temporal unit are unified according to the relations between intervals. Unification is represented by the relation `temp_unify(Term1,Term2,RUTerm)`, where its meaning is `RUTerm` is the temporal unification of `Term1` and `Term2` if

1. `Term1` *and* `Term2` *are equal terms, and* `Term1` *can be reduced to* `RUTerm`.
2. `Term1` *is not equal to* `Term2`, *and* `Term1` *can be reduced to* `RTerm1`, *and* `Term2` *can be reduced to* `RTerm2`, *and* `UTerm` *can be reduced to* `RUTerm`, *and the unification of the units* `RTerm1` *and* `RTerm2` *is* `RUTerm`.

The reduction of one term `reduce(Term,RTerm)` is true if `Term`

1. *is a temporal variable, given by* `time_var`, *and* `RTerm` *is equal to* `Term`. The `time_var` interpretation is a term is a temporal variable if it is a Prolog variable, or a canonical form of time unit where all its elements are Prolog variables.
2. *is a linear interval* `T1...T2`, *and* `T1` *can be reduced to* `RT1`, *and* `T2` *can be reduced to* `RT2`, *and* `RTerm` *is equal to* `RT1...RT2`;
3. *is a cyclical interval* `i(Range,Class)`, *and RTerm equal to* `Term`.
4. *is of the form* `Delta after T`, *and* `T` *can be reduced to* `TExp1`, *and* `RTerm` *can be reduced to* `TExp2`, *and the relation* `future(TExp1,Delta,TExp2) is true`. *The* `future/3` *relation means that the future of* `TExp1` *after the period of time* `Delta` *is* `TExp2`.

For instance,
`period(1,week) after (period(1,week) plus period(3,day) after t(14,2,1994)` is reduced to `t(1,3,1994)`.

Example 3. Suppose we have the following sentences in a knowledge base

```
foo(corn) @ i(5...6,month).
foo(tea) @ i(2...10, month).
foo(coffee) @ i(8...4,month).
foo(rice) @ I <== (foo(tea) & foo(coffee)) @ I.
```

In what follows, the symbols ":|" and ">>" represent the query and answer prompts, respectively, of the **NatureTime** system. When a query is done the dialog interface call the `solve/2`, and show only the second argument of it as the answer.

```
:| foo(corn) @ i(6...9,month).
>> foo(corn) @ i(6...6,month)
:| foo(corn) @ i(3...5,month).
>> foo(corn) @ i(5...5,month)
:| more.
>> Sorry, no further solution is possible.
|: foo(rice) @ T.
>> foo(rice)@i(8...10,month)
|: more.
>> foo(rice)@i(2...4,month)
|: more.
A:  Sorry, no further solution is possible!
```

The `more` command is just a special command of **NatureTime** to allow all possible solutions by backtracking (analogous to the ";" command in standard Prolog).

4 Application to Simulation Models

This section shows a direct application of our logic language to develop a simulation model for the growing process of the trees of the example of Sect. 2.1. After this, we point out the drawbacks of this implementation and of the language.

4.1 Simulation Models in NatureTime

In our previous scenario, as the scale of time use was week, we could define another MTC within the hierarchy defined, that is `mod_temp_class(week,day,7)`[3]. The rainy season, assumed to be only in February, can be written as `season(rain) @ interval(2...2,month)`. So, in every instance of February, within a MTC of year it will rain.

The height of a tree can be calculated by using the Equation (1), but we have to make the growth rate r_i a function of the interaction between the tree and its neighbour trees as in Equation (2). To do this, we first define a predicate to represent the neighbours of a tree along with the distance between them. The height must be calculated for each time step. This will be represented by the following predicate definition, where the base case is the initial height for each tree.

```
height(Tree,H) @ p(1,week) after T
  <==
   height(Tree,H1) at T                          &
   max_height(Tree,MAX)                          &
   real_gr(Tree,T...(p(1,week) after T,RGr)      &
   neighbours(Tree,NTrees)                       &
   influences(NTrees,T,R)                        &
   sum(R,TR)                                     &
   (Gr is RGr - TR)                              &
   (C is Gr*H1*(1 - H1/MAX))                     &
   (H is H1 + C).
```

The `influences/3` predicate represents the influence of the neighbours of a tree *since* the time `T`. The `sum/2` predicate represent the relation between a list of values and a number which is the sum of these values. The `real_gr/3` gets the supposed "real" growth rate of the tree throughout the interval of one week. The `real_gr/3` can be defined in a standard Prolog style, for example, as follows

```
real_gr(Tree,T,Gr):-                    % Gr is the growth rate if
   solve(season(rain) @ T,_),           % T is a rainy season
   growth_rate(Tree,LowGr),             % Lowgr is a standard Gr
   Gr is 1.2*LowGr.                     % Gr = LowGr*1.2
real_gr(Tree,T,Gr):-                    % Gr is the growth rate if
   \+ solve(season(rain) @ T,_),        % it is not true that T is
   growth_rate(Tree,Gr).                % a rainy season, and get Gr
```

Note that we call the `solve` meta-interpreter to use its facilities of unification as we saw in Sect. 3.3. The complete knowledge base of our example is shown in the Appendix A.

Below, we show the simulation of the growing process. Note that when the simulation "leaves" the season of raining, the growth rate decreases, as expected.

```
|: height(t1,H) @ T.
>> height(t1,1)@ t(14,2,1994)
```

[3] In a further extension of the logic we are going to treat the nested notations over MTC which are not part of the main hierarchy of time.

```
|: more.
>> height(t1,1.0097385444743936)@ t(21,2,1994)
|: more.
>> height(t1,1.019549379017451)@ t(28,2,1994)
|: more.
>> height(t1,1.0294326000485676)@ t(5,3,1994)
|: more.
>> height(t1,1.0376322103100148)@ t(12,3,1994)
|: more.
>> height(t1,1.0458800768343346)@ t(19,3,1994)
|: more.
>> height(t1,1.054176211878584)@ t(26,3,1994)
|: more.
>> height(t1,1.0625206230928805)@ t(3,4,1994)
```

4.2 Drawbacks of the NatureTime System

These include

- The limitation of the meta-interpreter which does not allow an explicit specification of temporal aspects of certain processes, e.g. the growth rate.
- The unification over the `after` operator causes a serious problem of time consumption. The reason is because recursive specifications on time steps produce big pure temporal expressions to be reduced. To be more precise, in the kernel of our unification algorithm, every time an expression of the form `P after T` is found, `T` is *reduced* to a canonical form, but it can also be another `P' after T'`, which is not canonical, and the reduction is applied, and so on.
  The problem with this is not the reduction algorithm itself, but the redundant recomputations of such pure temporal expressions. This could be solved as discussed in [Robertson *et al.* 91], if we had used some technique of recording the results of satisfying particular goals.
- The representation of different agents, e.g. forest, working at different time granularity would cause an exponential explosion in the search space of the deduction process. For instance, to produce information for forest in order to compute its influence in low levels, it would be necessary to regenerate this information.

All of these problems are much more related to the sequential computation of the logic-language used than to our theory of time granularity itself. Another way to attack the problem may help us not only to obtain efficient specifications, but also to achieve better integration of simulation models. We believe that our *Linear-Cyclic Hierarchy* of time is suitable for this new approach which we will briefly point out in the next section.

5 Ecological Simulation in Multi-Agent Systems: Some Insights

From the representational point of view, simulation models in ecology, developed in programming languages where the computation is sequential, do not feel particularly natural, mainly because, in such kinds of systems, many processes happen concurrently

in time, and ecological entities have their own behaviour (in a particular time scale), constantly reacting to the influence of their environment[Levin 92]. In this way, we propose to consider an eco-system as a very complex reactive system, in the sense of [Pnueli 86], since the behaviour of ecological entities are represented by simulation models. In this Section we address some insights about the *ecological agent* view of eco-systems, and we also suggest our time granularity theory as a basis for an executable temporal logic for simulation models in ecology.

5.1 From Logic-based to Agent-based Simulation

In the logic-based simulation of the Sect. 4, the clauses for the growth rate and maximum height of a tree can be seen as "individual knowledge" of each tree, and the clause for height as *common hierarchical knowledge* (CHK) for all species of tree, since the growing process of all trees are assumed to be same. In this way we may consider the previous set of clauses as a collection of agents. This view, similar to that of agent as part of expert systems as described in [Dieng 91], we say is a *weak agent definition*.

The clause for forest biomass can been seen as the knowledge from all agents, or the forest if we consider it as an agent as well. Our knowledge base would consist of a set of sets of clauses, each set for one agent, in a hierarchical way. For instance, in the previous example the biomass of a forest depends on the biomass of the trees, and the biomass of one tree at a given time depends on the its specific weight and its height, and its height depends on the influence of its neighbour trees at previous time. However, nothing in the language of specification is related with the view of the world as consisting of active agents.

The paradigm of Multi-Agent Systems (MAS) seems to fit quite well as a framework of modelling ecological systems. By using MAS, the behaviour of complex eco-systems is expected to emerge from the interactions between agents. As a consequence, integrations with higher accuracies can be done, and improved comprehension of how processes working at different levels of time granularity can be achieved. In this way, we propose to treat each ecological entity as an *ecological agent*, with its *individual knowledge* (internal and external) to represent its own state descriptions. We will need to specify the external description of an agent. This means which information about one agent is worth making available to other agents of its environment. We will also need to represent CHK about the relation of its processes with other species and the environment (some kind of "social reasoning").

The idea now is that instead of using a linear point structure of time, we can use the Linear-Cyclic structure of time, as defined in Sect. 3, because:

- it allows us to easily define as many levels of time granularity as needed. Furthermore, each agent could have its on clock, and we need to synchronise the clocks of different agents when integrating, or creating macro-agents by composing small ones.
- it allows us to define temporal cycles, also important in such simulation models.

By using this theory, we can assume that the time stamp `T` of the specification above can be in one of the time units [4], e.g. `year(3)` for the third year. In this way, assuming a time hierarchy with day, month, and year, the value of the tree's height for

[4] In the current extension of our time ontology we are introducing mechanisms to represent entities of all levels without using the representation $t(t_1, \ldots, t_k)$.

a given time could be "asked", for instance, `value(height,tree(t1),H) at month(3)`, which means "the value of the height of the tree t1 at the 3rd month of life is H".

To allow the definition of the time scale of the processes of an agent, the CHK of a group of agents related to a process, should contain the following predicate

`scale(time,tree,Process,MClass)`,

where `Process` is to say which process of the tree works in the level of time defined by the modular temporal class `MClass`. We also need to say which attribute of an agent is changed by a process. This can be defined by another predicate which is `change(Attribute,Process)`. As an example, the definition of a tree as an agent could be

```
Agent      : tree
Attributes: height, growth_rate, area_of_leaves,
            length_of_roots, spatial_position, specific_weight
Process    : growing, water_up_take, nitrogen_up_take, photosynthesis
Time Scale: scale(time,tree,growing,week).
            scale(time,tree,water_up_take,hour).
            scale(time,tree,nitrogen_up_take,hour).
            scale(time,tree,photosynthesis,minute).
Relation between processes and attributes
            change(height,growing).
            change(growth_rate,water_up_take).
            change(growth_rate,nitrogen_up_take).
            change(growth_rate,photosynthesis).
            ...
```

Based on a such high level of description, the time theory we provide may be considered with no commitment to the computational features of the model. However, due to the inefficiency of computation of systems like **NatureTime**, the theory may not be useful or provide mechanisms to allow complex models to be implemented. Furthermore, the concurrent aspect of eco-systems could be better understood if we had a language to specify the behaviour of ecological agents, i.e. the processes related with such agents. These processes may work at different scales of time and also in parallel. For instance, water and nitrogen up take, and photosynthesis may happen at the same time, with their effect noted in different time scales.

The efficiency of distributed systems as used in MAS is what we intend to use, and explore how to fit our time granularity theory, where each agent "acts" according to its own time scale. Now the problem is how to provide the logic with mechanisms which deal with different time granularities.

5.2 What Would be Needed?

To our knowledge, temporal logics for reactive systems developed so far do not worry much about processes working at different levels of time granularity. Our suggestion is that the use of an executable temporal logic, to specify simulation models in ecology, could help to make the points of contact more obvious, and the problem of integration more tractable. However, to our knowledge, the state of art in programming language for MAS is much more concerned with the tense aspects of agent's properties and behaviour through time than with the "durational" aspects of these features, e.g.

[Fisher 94]. One approach which seems to worry about it is AGENT0 [Shoham 90], but this does not deal explicitly with time granularity.

An executable temporal logic for the specification of simulation models for eco-systems, in the sense of MetateM [Fisher 94] but with time-stamped temporal operators, should have the following features.

- the temporal operators should be consistent in all levels of time granularity.
- the processes of transition, in any level, from one state to another should be represented by interaction with other processes occurring in other levels as well. For instance, if we want to introduce soil as an "agent" to control the resource of water, then it should be done by message passing from a tree to the soil.

In this way, each process related with the behaviour of an agent could be separately specified at different time scale. This means that for each agent there exists a set of "reactive systems", working and interacting concurrently in time, and at different levels of time granularity. We have implemented a very specialised MAS for the *Example 1*, by using Prolog as specification language, and the *tuple space* of Linda [Carriero & Gelernter 89] to represent the external description of agents and also as medium for passing of message. However, this does not completely correspond to what we really want to do, and to explore our time granularity theory within this architecture is our next step.

6 Concluding Remarks

In this paper we presented a new theory of time granularity which can be easily understood and used to define as many levels of time hierarchy as needed. We also showed that such a theory is useful in the representation of cyclical processes in simulation models for eco-systems.

Although this theory has been implemented in a very simple temporal reasoning system, we suggested that it can be used as a basis to develop a time-stamped executable temporal logic for the specification of simulation models, by involving agents working at different scales of time. This language should be based on a non-sequential computation in order to better represent agents interacting and concurrently working.

By using an executable temporal logic with such features, a modeller could specify his/her simulation model as an agent, with its own knowledge and scale of time. In this way, complex simulations would be expected to result from the interaction among different ecological agents.

Acknowledgements - The first author would like to thank Wamberto Wasconcelos, for helpful discussions on the time theory during our minutes for "academic coffee". We would also like to thank Robert Muetzelfeldt and Paulo Salles for many discussions on the stuff of ecological simulation.

References

[Allen & Hayes 85] James F. Allen and P. J. Hayes. A common sense theory of time. In *Proceedings of the of the IJCAI*, pages 528–531, 1985.

[Allen 83] James F. Allen. Maintaining knowledge about temporal intervals. *Communications of the ACM*, 26(11), 1983.

[Allen 84] James F. Allen. Towards a general theory of action and time. *Artificial Intelligence*, 23(1), 1984.

[Badaloni & Berati 94] Silvana Badaloni and Marina Berati. Dealing with time granularity in a temporal planning system. In *First International Conference on Temporal Logic*, pages 101–116, Bonn, Germany, July 1994. Springer-Verlag.

[Barringer *et al.* 91] Howard Barringer, Michael Fisher, Dov Gabbay, and Anthony Hunter. Meta-reasoning in executable temporal logic. In *Knowldege Representation*, 1991.

[Biggs 87] Norman L. Biggs. *Discrete Mathematics*. Oxford Science Publication, 1987.

[Carriero & Gelernter 89] Nicholas Carriero and David Gelernter. Linda in context. *Communications of ACM*, 32(4), 1989.

[Dean 89] Thomas Dean. Using temoral hierarchies to efficietly maintain large temporal databases. *Journal of the ACM*, 36(4):687–718, 1989.

[Dieng 91] Rose Dieng. Relations linking cooperating agents. In Yves Demazeu and Jean-Pierre Muller, editors, *DECENTRALIZED A. I. - 2*, pages 51–69. Elsevier Science Publishers, 1991.

[Fisher 94] M. Fisher. A survey of concurrent metatem - the language and its applications. In *First International Conference on Temporal Logic(ICTL)*, Bonn, Germany, July 1994.

[Fruhwirth 94] Thom Fruhwirth. Annotated constraint logic programming applied to temporal reasoning. Technical Report 22, ECRC, July 1994.

[Gabbay 87] Dov Gabbay. *Modal and Temporal Logic Programming. In Temporal Logics and their Applications - Antony Galton Editor.* ACADEMIC PRESS, 1987.

[Gabbay 89] Dov Gabbay. The declarative past and imperative future. In H. Barringer, editor, *Coloquim on Temporal Logics and Specification*, pages 409–448. Springer Verlag, 1989. V(398).

[Gray 93] Babak Gray. T-prolog - a temporal logic for an ecological domain. Unpublished M.Sc. thesis, Department of Artificial Intelligence/University of Edinburgh, 1993.

[Hobbs 85] Jerry R. Hobbs. Granularity. In *Proceedings of the of the IJCAI*, pages 1–4, 1985.

[Koomen 89] Johaness A.G.M. Koomen. Reasoning about recurrence. Technical Report Technical Report 307, Department of Computer Science - University of Rochester, Rochester, USA, 1989.

[Ladkin 86] Peter Ladkin. Primitives units for time specification. In *Proceedings of the AAAI*, pages 354–359, 1986.

[Leban *et al.* 86] Bruce Leban, David D. McDonald, and David R. Foster. A representation for collections of temporal intervals. In *Proceedings of the AAAI*, pages 367–371, 1986.

[Levin 92] Simon A. Levin. The problem of pattern and scale in ecology. *Ecology*, 73(6):1943–1967, 1992.

[Montanari 94] Angelo Montanari. A metric and layered temporal logic for time granularity, syncrony and asyncrony. Unplished work of the First International Conference on Temporal Logic, July 1994.

[Mota 94] Edjard Mota. Temporal representation of ecological domain. DAI Technical paper 31, Detartment of Artificial Intelligence/University of Edinburgh, 1994.

[Pnueli 86] Amir Pnueli. Specification and development of reactive systems. *Information Processing*, 1(1), 1986.

[Robertson *et al.* 91] David Robertson, Alan Bundy, Robert Muetzefeldt, Mandy Haggith, and Michael Uschold. *Eco-Logic Logic-Based Approaches to Ecological Modelling*. The MIT Press, 1991.

[Shoham 90] Yoav Shoham. Agent-oriented programming. Technical Report STAN-CS-90-1335, Stanford University - Department of Computer Science, Standford, California, 1990.

[Sterling & Shapiro 86] Leon Sterling and Ehud Shapiro. *The Art of Prolog*. The MIT Press, 1986.

A KB Definition for the Tree's Height

```
% ------------------------------------------------------------------
%                    NEIGHBOURS DEFINITION
neighbours(t1,[(t2,4.24),(t3,4.24),(t5,6)]).
neighbours(t2,[(t1,4.24),(t3,6),(t4,4.24),(t5,4.24)]).
neighbours(t3,[(t1,4.24),(t2,6),(t5,4.24),(t6,4.24),(t7,6)]).
neighbours(t4,[(t2,4.24),(t5,6),(t8,4.24),(t9,3)]).
neighbours(t5,[(t1,6),(t2,4.24),(t3,4.24),(t4,6),(t6,6),(t7,4.24),(t8,4.24)]).
neighbours(t6,[(t3,4.24),(t5,6),(t7,4.24)]).
neighbours(t7,[(t3,6),(t5,4.24),(t6,4.24),(t8,6)]).
neighbours(t8,[(t2,6),(t4,4.24),(t5,4.24),(t7,6),(t9,3),(t10,3)]).
neighbours(t9,[(t4,3),(t8,3),(t10,2.24)]).
neighbours(t10,[(t8,3),(t9,4.24)]).
% ------------------------------------------------------------------
%                  INITIAL HEIGHT OF EACH TREE
height(t1,1) @ t(14,2,1994).                % initial height of t1
height(t2,1) @ t(14,2,1994).                % initial height of t2
height(t3,1) @ t(14,2,1994).                % initial height of t3
height(t4,1) @ t(14,2,1994).                % initial height of t4
height(t5,1) @ t(14,2,1994).                % initial height of t5
height(t6,1) @ t(14,2,1994).                % initial height of t6
height(t7,1) @ t(14,2,1994).                % initial height of t7
height(t8,1) @ t(14,2,1994).                % initial height of t8
height(t9,1) @ t(14,2,1994).                % initial height of t9
height(t10,1) @ t(14,2,1994).               % initial height of t10
% ------------------------------------------------------------------
%                  MAXIMUM HEIGHT OF EACH TREE
max_height(t1,7).                           % max height of t1
max_height(t2,10).                          % max height of t2
max_height(t3,12).                          % max height of t3
```

```
max_height(t4,5).                          % max height of t4
max_height(t5,5).                          % max height of t5
max_height(t6,7).                          % max height of t6
max_height(t7,12).                         % max height of t7
max_height(t8,6).                          % max height of t8
max_height(t9,4).                          % max height of t9
max_height(t10,13).                        % max height of t10
% ------------------------------------------------------------------
%          STANDARD GROWTH RATE OF OF EACH TREE
growth_rate(t1,0.01).                      % rate growth of t1
growth_rate(t2,0.012).                     % rate growth of t2
growth_rate(t3,0.015).                     % rate growth of t3
growth_rate(t4,0.0009).                    % rate growth of t4
growth_rate(t5,0.008).                     % rate growth of t5
growth_rate(t6,0.011).                     % rate growth of t6
growth_rate(t7,0.015).                     % rate growth of t7
growth_rate(t8,0.01).                      % rate growth of t8
growth_rate(t9,0.006).                     % rate growth of t9
growth_rate(t10,0.018).                    % rate growth of t10
% ------------------------------------------------------------------
%          INFLUENCE OF OTHER TREES IN ONE TREE
influences([],_,[]).                       % 0 tree has no influence
influences([Y|T],Time,[R1|TR]):-           % the influence of each
  ind_influence(Y,Time,R1),                % individual tree and
  influences(T,Time,TR).                   % the influence of rest

ind_influence((Tree,D),T,I):-              % Tree's influence is I
  solve(height(Tree,H) @ T,_),             % H is Tree's height @ T
  (I is H/(D*1000)), !.                    % I = H/(D*1000)).
```

This article was processed using the LaTeX macro package with LLNCS style

REPRESENTING INTERACTION OF AGENTS AT DIFFERENT TIME GRANULARITIES

MOTA, E.; ROBERTSON, D.

DAI Research Paper No. 796
Mar 1996

In the proceedings of TIME'96, 3rd International Workshop on Temporal Representation and Reasoning, Florida, 19-20 May.

Representing Interaction of Agents at Different Time Granularities*

Edjard Mota & David Robertson

Department of Artificial Intelligence
The University of Edinburgh
80 South Bridge, Edinburgh EH1 1HN Scotland
email: {edjardm,dr}@aisb.ed.ac.uk

May 15, 1996

Abstract

In this paper we describe **NatureTime** *logic which we use to represent and reason about the behaviour of interacting agents (in an ecological domain), which behave at different time granularities. Although the traditional application fields of temporal representation and reasoning still raise many interesting theoretical issues, we have been investigating some practical problems of ecological systems which suit different representations of time than those embodied in traditional simulation models of ecosystems. These seem well suited to reconstruction using temporal logic programs.*

1 Introduction

An understanding of the world generally needs a way to represent processes at different levels of granularity [1]. This may be done in relation to time, space, and the structural organisation of things we are interested in observing, representing and in many cases simulating. In relation to time and space, it is common that that processes working at finer levels of time can only be observed if we also change to a finer level of space, in order to observe the changes they may produce in the environment. An interesting aspect of these levels of abstraction is to observe how things, which happen at the lowest levels will affect the others at higher levels of granularity. This process is usually called scaling up. Things become complicated when the higher levels also affect, at longer periods of time, the lower levels.

*This work and the first author (on leave for PhD from the Department of Computer Science of the University of Amazonas, Manaus, Brazil) are sponsored by the Brazilian Ministry of Education, grant nº 01723/93-8/CAPES.

The scaling up process is highly relevant to the development of of simulation models of ecosystems. Many factors make scaling difficult for ecological modellers but a major obstacle is that each individual model is devised to run on its own, single-level, time scale. It is argued in [2] that the use of a temporal logic-based language to specify such simulation models as agents can provide better representations for the behaviour and interaction of eco-agents working at different scales of time. The usefulness of using **NatureTime** logic for the specification of such agents has already been shown [3].

In this paper we present some results on the representation of interacting agents developed in [4], which extends the **NatureTime** logic. We also address some new problems to be tackled when asynchronous agents are introduced into the environment where there are other interacting objects.

In Section 2, we present one example as a motivation for this work, and we also address some related works. In Section 3, we make a brief presentation of the **NatureTime** logic. In Section 4, we show how to enhance the logic with mechanisms for the representation of interacting eco-agents. In Section 5, we show an application of **NatureTime** to the specification and simulation of eco-agents working at different time granularities. Finally in Section 6 we give some concluding remarks.

2 Motivation and Related Works

2.1 Two Interacting Ecological Agents

Example: "We have a model of tree growth, expressed on a weekly time scale. This must interact with a model of a insect pest which moves up and down the tree on a daily time scale. The tree has its growth rate reduced

by 0.02 every day the pest moves above 8 meters. The pest moves continuously up and down, at a rate of 2 meters per day, reversing direction when it reaches the top or bottom".

This example shows an interaction between two entities working at different time scales. The types of question that we might want to answer in problems like this are: "what is the value of the attribute of each agent at an specific time?" or " when will some attribute of either agent have a certain fixed value?".

The behaviour of each of these agents could be represented by means of differential equations, which is an expressive way of representing the *continuity* of their behaviour. But a continuous representation of time is not always best for ecological models. For example, if we want to represent the behaviour of agents which immigrate and emigrate from one population to another, then it is difficult to represent this using continuous functions. A more usual way of modelling the changes in the state of such agents is to perform them at the time step of their corresponding granularity. This yields a discrete approximation to continuous models.

However, a discrete approach does not allow us to compute the value of some attribute at a time in between two consecutive time steps. To overcome this, interpolation could be needed. In this case we have to assume that there is no interaction between the agents that may affect the attributes in question. Otherwise, we cannot estimate the next value in the consecutive time step. Such a future value will depend on the value we are trying to find for the time in between the future and the past. In this work we just take the value of the most recent temporal entity in the past.

2.2 Related Work

Granularity is very important if we intend to look at the world at different levels of abstraction, when switching from one level to another may be necessary for the comprehension of the phenomena being observed [1]. A theoretical analysis of hierarchical time intervals was proposed in [5], where an elaboration over the interval calculus [6] is done to achieve a time framework where units of time can be specified. This work extends the idea of *convex* to *union-of-convex* intervals [7], where there may exist gaps between *convex* intervals. This allows the representation of collections of intervals and limited expressions of cyclicity. However, it was not implemented, as far as we know, so the pragmatics of using it for computation are obscure.

A similar approach was proposed in [8], where the basic idea in this approach is to use a set of primitive collections to specify other collections by using two operators, *slicing* and *dicing*, in order to select intervals from collections of intervals. Each such a primitive is defined by specifying the intervals of which it is composed. In this approach, circular aspects of time can be obtained from the *δ-values* which are treated as if they were a circular list. Although this approach was shown to be useful for reasoning about scheduling, it does not deal with different granularities of time.

In [9, 10] there is proposed a many-sorted first order logic augmented with temporal operators and a metric on time to deal with time granularity. This is achieved by introducing *contextual* and *projection* operations into topological logic [11]. This has been applied in the context of planning systems [12], to achieve plan actions at different scales of time and reduce the computational complexity of such systems.

In a parallel work to ours, [13] proposes an interesting framework of time based on the notion of *calendars* as being cyclic temporal objects. The difference is in the way in which such temporal objects are conceived. This approach, as the others does not include the cyclical aspect of time in their models.

The theory developed in [14] was an attempt to provide a logic based language to represent concepts of time, following closely the forms of expression used informally in descriptions of ecological systems in a target domain. Many of these descriptions include the idea of cyclical processes at many levels of time granularity. In the next section we will summarise the time theory developed to deal with these.

3 A Linear-Cyclic Hierarchy of Time

In this section we will briefly present the time hierarchy of the **NatureTime**. We refer to [14] for more details about the temporal reasoning interpreter, which is basically a standard Prolog meta-interpreter with restricted forms of unification on temporal labels.

3.1 Basic Assumptions

By analysing the temporal knowledge about seasonal cycles of ecological knowledge [15], as in sentences like "Coffee is harvested from August up to April", a hierarchy of time cycles was proposed in [14] in order to represent and reason about cyclical events. A natural mathematical structure for representing cycles is modular arithmetic (also called clock arithmetic [16]). For instance, years are modular sets of months, [lunar] months are sets of days, and so on. Note that this hierarchy of cycles also allows us to define many levels of temporal granularity.

3.2 Elements of the Language

The language is Prolog based enhanced with some special symbols, terms, and a unification algorithm for

temporal labels, which also works over terms of second order which not include predicate symbols.

Vocabulary - It is formed by symbols for variables, constants, functions and predicates as follows.

- two countably infinite sets of variables $\mathcal{L}_v$, and $\mathcal{L}_{tv}$, where x_i, y_i, z_i are variables of $\mathcal{L}_v$, s_i, t_i, u_i are variables of $\mathcal{L}_{tv}$.
- a finite set $\mathcal{L}_c$ of constants.
- a finite set $\mathcal{L}_f$ of non-temporal function symbols of the form f^n, where n is the arity of the function.
- a finite set $\mathcal{L}_{tc}$ of temporal constants defined as $\{lowest, flowtime, infinity, smallest\} \cup \mathcal{T}_C$, where $\mathcal{T}_C = \{c_1, \ldots, c_n\}$, and each c_i is a special constant or names of temporal classes.
- a finite set of temporal function symbols f_t^n, where n is the arity of the function, $\mathcal{L}_{tf} = \{p^2, i^2, t^n, ...^2, plus^2, after^2, before^2, of^2\}$, where $...^2$, $plus^2$, $after^2$, $before^2$, of^2 are all of arity 2 but written using infix notation.
- a finite set $\mathcal{L}_p$ of predicate symbols p_i^n, $i \geq 1$, where $n > 0$ is the arity of p_i, where *mod_temp_class* is special predicate symbol of arity 3, and *on* is a special predicate of arity 2.
- the set $\mathbb{Z}$ of integers is also part of the vocabulary.
- the propositional connectives for conjunction, disjunction, and implication are represented here by &, $\vee$, and $\Longleftarrow$, respectively. The truth value for true is represented by $\top$, and false by $\bot$.
- the temporal symbol "@".

Classes of Expressions - By using these symbols we define the following classes of expressions where the capital letters A, B, C are used for formulae.

. a *logical term* is recursively defined as
  - a variable $x \in \mathcal{L}_v$ is a term
  - a constant $c \in \mathcal{L}_c$
  - if $t_1, \ldots, t_n$ are terms and $f \in \mathcal{L}_f$ and has arity n, then $f^n(t_1, \ldots, t_n)$ is a term.

Examples of logical terms are *maize*, *grass*, $height(tree(t_1))$, $biomass(forest(f_1))$.

. a *temporal term* (TT) is a temporal variable $s \in \mathcal{L}_{tv}$, or a temporal constant $c_t \in C_T$, or a temporal function symbol $f_t \in \mathcal{L}_{ft}$ in one of the following forms.
  - a *period*, recursively defined as a 1) *single period* $p(s, m)$ where $s \in \mathbb{Z}$ and $m \in \mathcal{T}_C$; 2) a *composite period* P *plus* P', where P is *single period*, and P' is a *period* term.
  - a *cyclical interval* $i(s...t, m)$, where s, $t \in \mathbb{Z}^+$ and $m \in \mathcal{T}_C$
  - a *smallest temporal entity* $t(t_1, \ldots, t_k)$, where for n as the number of elements in $\mathcal{T}_C$, then $k \leq n$, each $t_i \in \mathbb{Z}^+$, and each i correspond to exactly one element of $\mathcal{T}_C$.
  - a *linear interval* $s_1...s_2$, where s_1 and s_2 are in the form $t(t_1, \ldots, t_k)$.
  - a *collection interval*
    * α where $\alpha \in \mathcal{T}_C$.
    * $f_t(n)$, where $f_t \in \mathcal{L}_{tf}$, and $n \in \mathbb{Z}^+$, and f_t corresponds to one unique symbol $\alpha \in \mathcal{T}_C$. For instance, the function symbol of $week(1)$ corresponds to a *week* of $\mathcal{T}_C$
    * $\alpha(n)$ *of* S, where $f_t \in \mathcal{L}_{tf}$ (same as previous item), and $n \in \mathbb{Z}^+$, and S is a *collection interval*.

Examples of TT are, for example, $i(11...5, month)$ to mean *interval of time between November and May*, $p(1, day)$ *after* $t(17, 7, 1994)$ *one day after 17th July, 1994*, $day(1)$ *of week* means *all Mondays*, and $last(day(2)$ *of week of* $month(2)$ *of* $year(1996))$ means *the last Monday of February of 1996*. The *collection interval* is more general than a *cyclical interval* in the sense that it may define cyclical and noncyclical intervals. For instance, $day(2)$ *of week of month* represents all Mondays of all months. However, $day(2)$ *of week of* $month(2)$ *of* $year(1996)$ is not cyclical because it represents the sequence of days 5th,12th, 19th, and 26th of February 1996.

. a pure temporal expression (PTE)
  - t, if t is a TT, but not a *period term*.
  - p *after* t, where p is a *period* term, and and t is a pure temporal expression.

Examples of PTE are $p(3, month)$ *after* $i(10...11, month)$, $p(24, year)$ *after* $t(17, 7, 1970)$, $hour(4)$ *of* $day(1)$ *of* $week(1)$ *of* $month(3)$.

. an atomic formula (AF) is

  - $\top$ and $\bot$ are atomic formulae
  - if $term_1, \ldots, term_n$ are logical terms and $p^n \in \mathcal{L}_p$, then $p^n(term_1, \ldots, term_n)$ is an atomic formula

. classical atomic formulae can be annotated with a PTE by using the temporal operators "@", forming an atomic temporal formula (ATF), i.e.

  - A, if A is an AF, then it is an ATF.
  - A @ T, where A is an AF and T is a PTE, is an ATF

  For instance, $harvested(maize, highlands)$ @ $i(12...1, month)$ means that *maize is harvested throughout the whole interval of time from December up to January.*

. *body* is in one of the forms A & B, $A \vee B$, or C, where A and B are bodies and C an ATF. A typical example of a body is $time\_between(i(2...4, month), p(3, month))$ & $harvested(maize, high\_lands)$ @ $i(12...1, month)$.

. a well formed temporal formula (WFTF) is

  - if A is an ATF, then A is a WFTF with an *empty body*
  - if A is an ATF and $B \neq \top, \bot$, and B is a body then

    $A \Longleftarrow B$ is a WFTF, and A is called the *head*

  A WFTF where its *head* is an AF, and *body* is formed only by AF is a Prolog clause with no temporal contents. An example of a WFTF is

  $$harvested(Crop)@\ p(D, C)\ after\ T \Longleftarrow planted(Crop)\ @\ T.$$

  Which means *if a Crop is planted throughout and interval T, then it is harvested a D units C of time after T.*

3.3 The Linear-Cyclic Hierarchy Structure

The temporal structure used is a 5-tuple $\mathcal{LC}_H =_{def} \langle \mathcal{T}_h, \mathcal{E}, \overset{t}{\succ}, T, < \rangle$, where $\mathcal{T}_h$ is a finite partially ordered set of *modular temporal classes* (MTCs), $\mathcal{E}$ is a non empty set of temporal entities (or units of time), $\overset{t}{\succ}$ is a binary relation of temporal succession within the hierarchy (the properties of this relation are described in the Appendix A), T is the set of integers ordered by the binary relation of precedence $<$. The set $\mathcal{T}_h$ is composed of two sets $\mathcal{M}_h$ and $\mathcal{F}$ $\{mc_1, \ldots, mc_n\}$. The second is called *fluctuating* MTC, and the former is the main time hierarchy which is induced by the following relations.

. $mc_{n+1} =_{def} mod\_temp\_class(flow\_time, c_n, infinity)$, where $c_n \in \mathcal{T}_C$

. $mc_i =_{def} mod\_temp\_class(c_i, c_{i-1}, m_v)$, where $1 < i \leq n$, $c_i, c_{i-1} \in \mathcal{T}_C$, and $m_v \in \mathbb{Z}^+$

. $mc_1 =_{def} mod\_temp\_class(c_1, lowest, smallest)$, where $c_1 \in \mathcal{T}_C$. Each c_i is called the name of the class mc_i. The flow of time is a sequence of instances of c_n, where each one denotes a stage in a cycle and therefore can recur. For instance, a simple hierarchy which includes temporal classes for a simplified calendar model of time could be defined as follows.

$mod\_temp\_class(flow\_time, year, infinity)$.

$mod\_temp\_class(year, month, 12)$.

$mod\_temp\_class(month, day, 30)$.

$mod\_temp\_class(day, lowest\_level, smallest)$.

A view of this hierarchy is depicted in in Figure 1.

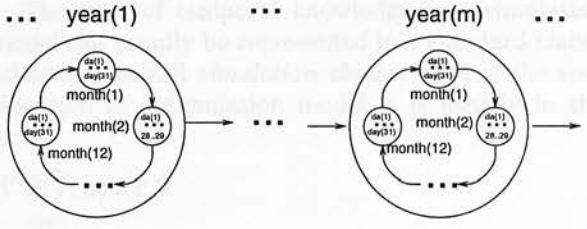

Figure 1: A view of the *Linear-Cyclic Hierarchy* of time. The inner circles represent lower levels of time granularity like *day* and *month*, while the outest circles represent the highest level, like *year*.

These relations define the flow of time, represented by $flow\_time$, as a linear and infinite ($infinity$) sequence of *years*, *year* as a MTC of 12 *month*, *month* as a MTC of 30 *days*, *day* as the smallest (*smallest*) time interval. Note that in this hierarchy we consider months as regular MTC, which is not the case in the real calendar. Although the logic allows us to define real calendars, we will omit it from this paper.

3.4 A Meta-Interpreter for NatureTime

In this section we present a meta-interpreter based on [17] for the **NatureTime** rather than a formal semantics. As we treat temporal reasoning as a problem of unifying PTEs, then we need to control the part of the unification which deals with it. Because of this, the standard $solve/1$ predicate is changed to be a binary relation $solve/2$. Basically, the meta-interpreter

accepts queries which are temporal formulae. It succeeds if the query holds throughout some interval within the given PTE. Since it may not hold across the entire interval it is necessary to also show what the restricted PTE is - hence the second argument. In this way we have the following meta-interpreter.

solve$(A$ @ $T1,\ A$ @ $T3) : -$
  $\neg A = (\_ \ \& \ \_)$,
  A @ $T2$,
  temp_unify$(T1, T2, T3)$.
solve$(A$ @ T, A @ $RT) : -$
  $A = (A1 \ \& \ A2)$,
  solve$(A1$ @ $\_, A1$ @ $T1)$,
  solve$(A2$ @ $\_, A2$ @ $T2)$,
  temp_unify$(T1, T2, RT1)$,
  temp_unify$(T, RT1, RT)$.
solve$(A \ \& \ B, A1 \ \& \ B1) : -$
  solve$(A, A1)$,
  solve$(B, B1)$.
solve$(A$ *or* $B, R) : -$
  (*solve*(A, R) ;
  solve(B, R)).
solve$(A$ @ $T1, A$ @ $T3) : -$
  A @ $T2 \Longleftarrow$ *Precondition*,
  solve$(Precondition, \_)$,
  temp_unify$(T1, T2, T3)$.

As clauses 3 and 4 are just another way to re-write the standard clauses for logical conjunction and disjunction, we assume their usual interpretation. We have the following interpretation.

1 - *A is true throughout T3, if A is not a composite event, and A is true throughout T2, and T3 is the temporal unification of T1 and T2.*

2 - *A is true throughout T if, A is a composite event consisting of A1 & A2, and A1 is true throughout T1, and A2 is true throughout T2, and RT1 is the temporal unification of T1 and T2, and RT is the temporal unification of T and RT1.*

5 - *A is true throughout T3 if, the A @ T2 is the head of a temporal Horn clause with body Precondition, and the body can be solved, and T3 is the temporal unification of T1 and T2.*

Briefly, the temporal unification concerns with the reduction of PTE to canonical forms of TT, and then unifying them by performing modular or linear matching using the usual relation between time intervals [6]. The TTs S and T unify if one of the following cases applies.

1. *$S = s_1...s_2$ and $T = t_1...t_2$, then either they overlap or one is included into the other.*
2. *$S = i(s_1 \ ... \ t_1, C)$ and $T = i(t_1... \ t_2, C)$, and mod_temp_class$(_, C, M)$ holds, M is an integer, and $i(s...t, C)$ is the modular match between S and T.*
3. *$S = i(s_1...s_2, C)$ and $T = t_1...t_2$, if there is one time instance $S' = s'_3...s'_2$ of S, and T and S' unify.*
4. *$T = t_1...t_2$ and $S = i(s_1...s_2, C)$, then S and T also unify.*

The *time_instance* creates one linear instance of a cyclical interval. As a cyclical interval represents a collection of linear intervals, there may exist many instances of it in the level of linear intervals.

4 NatureTime for Eco-Agent Specification

This section shows an extension of the meta-interpreter of the **NatureTime** for reasoning about the state of an agent at any time in between two consecutive time steps. To this, we take some advantage of a common way of representing simulation models.

4.1 Dealing with Simulation Clauses

The type of temporal knowledge in a simulation model can usually be represented in a standard clause schemata we call *simulation clause*. If A is the specification of a simulation model it is usually in the form

A' @ P *after* T
  $\Longleftarrow$
  A @ T &
  $\mathcal{R}(A, A')$.

where P is normally in the form $p(1, C)$ and C is a modular temporal class of the time hierarchy, and the predicate $\mathcal{R}$ is intended to represent the sequence of formulae which involves A and A' to produce their relationship within the flow of time. Finally, A and A' have the same predicate symbol and the same arity. Usually they are in the form *value*$(Attribute, Agent, V)$, i.e. the value of an *Agent*'s *Attribute* is V. In order to allow **NatureTime** to reason about those queries of 2, taking into account this particular type of clause, we have the following solution.

1. check if the given temporal entity is a fixed time
2. check if there is a simulation clause such that the formula matches with some part of its body, and that the proposition A has the same structure (i.e. the same predicate symbol and same number of arguments),

3. find one solution for $A1 \ @ \ Ti$, where Ti is a variable

4. if the solution found is the proposition searched within the the specified interval, then the search stops.

5. otherwise, it the PTE of the head of the simulation clause is the future of T then the last state of A is assumed to be the required value, and the search stops.

6. otherwise the constraints represented in the $\mathcal{R}$ relation are attempted to be solved, and the future state of $A1$ throughout interval Ti is the head of the simulation clause, and we back to step 4.

4.2 Representing Interaction Between Agents

As **NatureTime** offers mechanisms for specifying simulation models working at different levels of time granularity, and ecological systems usually involve entities acting on their own clocks, then an ecosystem modelled in such a logic is a straightforward representation of an agent's behaviour in the sense of multi-agent systems (MAS). Such a view is the same as in [18]. In what follows, we consider that agent's attributes which depend on processes specified at a certain scale of time will have their state changed only at that level.

If an agent A_i acting at a coarse level of time interacts with other A_j working at lower level, and the result of this interaction is that an attribute Att_j of A_j affects and attribute Att_i of A_i, then A_i has to find out the sequence of values for Att_i. For instance, the model of a tree for the example we gave should find out the sequence of values for the bug's position during the period of time in which its attribute is assumed to be constant.

The MAS approach we are using takes into account that an agent may "read", or observe the value of the attributes of other agents if such information is relevant to its behaviour. Thus each agent may need a process dedicated to obtaining such information. We will call this *observer process* (OP). In our case, the OP will get all the values of Att_j, in a list L, during an specific period of time of the length of L, and then the agent A_i will compute the influence of these values on its attribute Att_i. This idea is depicted in Figure 2.

Note that the OP needs to know the time scale of the agent A_j in order to know how to relate both scales of time. Thus, every agent must specify the scale of time of the internal processes responsible for

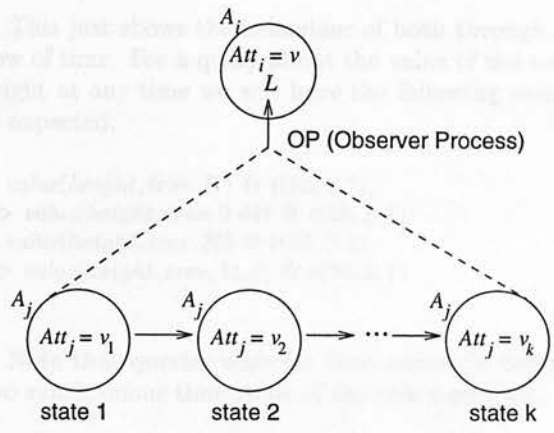

Figure 2: Interaction between agents A_i and A_j, where the OP of A_i get the list L of values of Att_j during a period k units of time.

its behaviour, and also a dependency relation between attribute and process. This will be done by using the predicates $scale(time, Agent, Process, C)$, where C is a MTC, and $depend(Attribute, Process)$.

In this way, we extend the language by introducing a builtin predicate which is specified in the level of the meta-interpreter. This is the predicate $progress/4$ for representing the progress observed of the value of an attribute Att of an agent Obj, from a given temporal entity T during a given period of time P, and the progress will return in a list. A simple specification of the $progress/4$ predicate can be, for example, as follows.

$$
\begin{array}{l}
progress(value(Att, Obj, V)@T, P, [V|R]) : - \\
\quad depend(Att, Proc), \\
\quad scale(time, Obj, Proc, C), \\
\quad solve(value(Att, Obj, V) \ @ \ T, \_), \\
\quad future(T, P, Tf), \\
\quad progression(value(Att, Obj, V) \ @ \ T, Tf, C, R).
\end{array}
$$

$$
\begin{array}{l}
progression(\_ \ @ \ T, Tf, C, [\,]) : - \\
\quad next(T, C, Tf). \\
progression(value(Att, Obj, Vi) \ @ \ Ti, Tf, C, [Vj|R]) : - \\
\quad \neg \ next(Ti, C, Tf), \\
\quad value(Att, Obj, Vj) \ @ \ p(1, C) \ after \ Ti \\
\qquad \Longleftarrow value(Att, Obj, Vi) \ @ \ Ti \ \& \ Constraints, \\
\quad solve(Constraints, \_), \\
\quad next(Ti, C, Tj), \\
\quad progression(value(Att, Obj, Vj) \ @ \ Tj, Tf, C, R).
\end{array}
$$

5 Representing the Tree and Bug Interaction

For the sake of simplicity, we will assume that both eco-agents start their behaviour at the same time. In this way we have the following facts in our KB.

$growth\_rate(tree, 0.5)$.
$scale(time, bug, movement, day)$.
$scale(time, tree, growing, week)$.
$depend(height, growing)$.
$value(height, tree, 6)$ @ $t(1,1,1)$.
$value(pos, bug, 6)$ @ $t(1,1,1)$.

The specification of the *tree*'s growing process can be, for example, as follows.

$value(height(tree), H)$ @ $p(1, week)$ $after$ T
$\Leftarrow$
$value(height, tree, Hi)$ @ T &
$progress(value(pos(bug, \_), \_)$ @ $T, p(1, week), L)$ &
$growth\_rate(tree, GR)$ &
$influence(GR, L, RealGR)$ &
$(HisHi + RealGR)$.

The specification of the *bug*'s movement can as follows.

$value(pos, bug, 6)$ @ $t(1,1,1)$.
$value(pos, bug, PB)$ @ $p(1, day)$ $after$ T
$\Leftarrow$
$value(pos, bug, PBi)$ @ T &
$value(height, tree, H)$ @ T &
$new\_pos(PBi, H, PB)$.
$new\_pos(Pos1, H, Pos2) : -$
$Pos1 \leq H$,
$Pos2$ is $Pos1 + 2$.
$new\_pos(Pos1, H, Pos2) : -$
$Pos1 > H$
$Pos2$ is $Pos1 + 2$

The $new\_pos/3$ simply changes the bug's position according to its behaviour as specified in the Example. For this specification we have the following results for the simulation of the bug's position.

$|: value(pos, bug, Pos)$ @ T.
$>> value(pos, bug, 6)$ @ $t(1,1,1)$
$|: more$.
$>> value(pos, bug, 8)$ @ $t(2,1,1)$
...
$>> value(pos, bug, 8)$ @ $t(8,1,1)$
$|: more$.

For the tree's growing process we have.

$|: value(height, tree, H)$ @ T.
$>> value(height, tree, 9)$ @ $t(1,1,1)$
$|: more$.
$>> value(height, tree, 9.44)$ @ $t(8,1,1)$
...
$>> value(height, tree, 11.2)$ @ $t(6,2,1)$
$|: more$.
$>> value(height, tree, 11.64)$ @ $t(13,2,1)$

This just shows the behaviour of both through the flow of time. For a query about the value of the tree's height at any time we will have the following results, as expected.

$|: value(height, tree, H)$ @ $t(10,1,1)$.
$>> value(height, tree, 9.44)$ @ $t(10,1,1)$
$|: value(height, tree, H)$ @ $t(10,2,1)$.
$>> value(height, tree, 11.2)$ @ $t(10,2,1)$

Note that queries were for time values in between two synchronous time steps of the tree's growing.

6 Concluding Remarks

In this section we make a brief analysis on the results obtained, the limitations still to be overcome.

Extension of **NatureTime** avoids re-computation of pure temporal expressions already computed by the recursive definition. To our knowledge this is a new technique for controlling the search taking into account a restrict form of clause.

A limitation of the $progress/3$ is that it is assumed that all processes working synchronously rather than asynchronously. This is depicted in Figure 3, where the lines for P_1 and P_2 represent their clock, and each mark is the time point in which the attributes they change are updated.

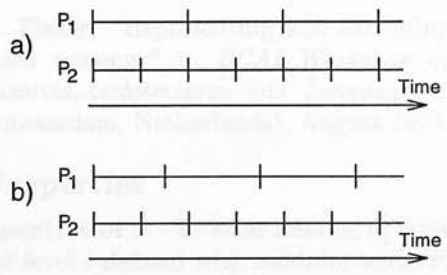

Figure 3: Interacting process P_1 and P_2 where in a) they are synchronous and in b) asynchronous.

For instance, if another agent is introduced at time $t(5,1,1)$ and needs to know the value of the tree every week, the PO for this agent would get the list [9] as the value of the height of the tree. However the correct list should be [9, 9.44] because the new agent was introduced asynchronously in relation to the other agents.

Although the use of the logic for the specification of more complex problems sometimes leads to inefficient local computations, the modularity of computational logic allows us to think in terms of distributing the computation over a set of agents, each one with its own time granularity. We have already done some

experiments [2] that strengthen our belief in the value of this type of architecture.

Acknowledgements - We would like to thank very much to Robert Muetzelfeldt for his contribution to our understanding on ecology. We also like to thanks Alan Smaill for his valuable contributions on the time theory. Finally, we thank to the patiency of the referees who carefully read and gave valuable comments for this final version.

References

[1] J. R. Hobbs, "Granularity," in *Proceedings of the of the IJCAI*, pp. 1–4, 1985.

[2] E. Mota, "Time granularity in simulation models within a multi-agent system." DAI Discussion Paper 158, Department of Artificial Intelligence, University of Edinburgh, April 1995.

[3] E. Mota, M. Haggith, A. Smaill, and D. Robertson, "Time granularity in simulation models of ecological systems," in *Workshop on Excutable Temporal Logics- Montreal, Canada*, DAI RP-740, Edinburgh University, 1995.

[4] E. Mota, D. Robertson, and R. Muezelfeldt, "On the granular aspects of time in simulation models," TP-39, Detartment of Artificial Intelligence/University of Edinburgh, 1995.

[5] P. Ladkin, "Primitives units for time specification," in *Proceedings of the AAAI*, pp. 354–359, 1986.

[6] J. F. Allen, "Maintaining knowledge about temporal intervals," *Communications of the ACM*, vol. 26, no. 11, 1983.

[7] P. Ladkin, "Time representation: A taxonomy of interval relations," in *Proceedings of the AAAI*, pp. 360–366, 1986.

[8] B. Leban, D. D. McDonald, and D. R. Foster, "A representation for collections of temporal intervals," in *Proceedings of the AAAI*, pp. 367–371, 1986.

[9] E. Ciapessoni, E. Corsetti, A. Montanari, and P. S. Pietro, "Embedding time granularity in a logical specification language for synchronous real-time systems," *Science of Computer Programming*, vol. 20, no. 1, pp. 141–171, 1993.

[10] A. Montanari, "A metric and layered temporal logic for time granularity, syncrony and asyncrony." Unplished work of the First International Conference on Temporal Logic, July 1994.

[11] N. Rescher and A. Urquhart, *Temporal Logic.* Springer Verlag, 1971.

[12] S. Badaloni and M. Berati, "Dealing with time granularity in a temporal planning system," in *First International Conference on Temporal Logic*, (Bonn, Germany), pp. 101–116, Springer-Verlag, July 1994.

[13] D. Cukierman and J. Delgrand, "A language to express time intervals and repetition," in *Proceedings of the of 2nd International Workshop on Temporal Representation and Reasoning*, (Melbourne Beach, Florida - USA), April 1995.

[14] E. Mota, "Temporal representation of ecological domains," DAI TP- 31, Department of Artificial Intelligence, University of Edinburgh, 1994.

[15] M. Haggith, D. Robertson, D. Walker, F. Sinclair, and R. Muetzelfeldt, "TEAK - tools for eliciting agroforestry knowledge," in *British Computer Society Symposium of IT - Enabled Charge in Developing Countries*, 1992.

[16] N. L. Biggs, *Discrete Mathematics.* Oxford Science Publication, 1987.

[17] L. Sterling and E. Shapiro, *The Art of Prolog.* The MIT Press, 1986.

[18] M. Fisher, "Representing and executing agent-based systems," in *ECAI Workshop on Agent Theories,Architectures and Languages (ATAL)*, (Amsterdam, Netherlands), August 1994.

A Properties

Properties of $\overset{t}{\succ}$ - In what follows, C_i^{mi} means the MTC of level i defined with modular value mi. This relation establish a sub-division relationship between MTCs. The $\overset{t}{\succ}$ relation has the following properties.

- *transitive* - if C_i^{mi}, C_j^{mj} and C_k^{mk} are MTCs and $C_k^{mk} \overset{t}{\succ} C_j^{mj}$ and $C_j^{mj} \overset{t}{\succ} C_i^{mi}$, then $C_k^{mk} \overset{t}{\succ} C_i^{mi}$.
- *reflexive* - if C_i^{mi} is a MTC then $C_i^{mi} \overset{t}{\succ} C_i^{mi}$ (every MTC can be subdivided in itself)
- *anti-symetric* - if C_i^{mi}, C_j^{mj} are MTCs, and $i \neq j$, and $C_j^{mj} \overset{t}{\succ} C_i^{mi}$, then $C_i^{mi} \overset{t}{\not\succ} C_j^{mj}$.

J. Symbolic Computation (1996) **22**, 665–698

NatureTime: Temporal Granularity in Simulation of Ecosystems†

EDJARD MOTA‡, DAVID ROBERTSON‡ AND ALAN SMAILL‡

Department of Artificial Intelligence, The University of Edinburgh
80 South Bridge, Edinburgh EH1 1HN, Scotland

(Received 31 October 1995)

Granularity of time is an important issue for the understanding of how actions performed at coarse levels of time interact with others, working at finer levels. However, it has not received much attention from most AI work on temporal logic. In simpler domains of application we may not need to consider it a problem but it becomes important in more complex domains, such as ecological modelling. In this domain, aggregation of processes working at different time granularities (and sometimes cyclically) is very difficult to achieve reliably. We have proposed a new time granularity theory based on *modular temporal classes*, and have developed a temporal reasoning system to specify cyclical processes of simulation models in ecology at many levels of time.

1. Introduction

Temporal logics and reasoning systems usually treat time as a linear sequence of discrete points or linear intervals. Such abstract views of time have been used in the development of many specification languages for real-time systems, databases, planning, etc. One might expect that temporal logics would be a natural way to represent conventional simulation models, since these also represent change over time. However, when we attempt to do this we almost immediately encounter obstacles: different parts of the model may operate at different temporal granularities; processes may operate cyclically; the axioms familiar to temporal logicians may be far removed in programming style from those of simulationists. We encountered all these obstacles when applying temporal logics to ecological modelling. Moreover, many phenomena in nature cannot be easily understood in only one scale of time, and using different scales is essential to their comprehension.

Nowadays, integration of ecological models is an important issue. There are many individual models of parts of systems, and people want to solve problems which require the behaviour of a number of different models to be combined. However, there is little

† This work and the first author (on leave from Department of Computer Science, University of Amazonas-Brazil) are sponsored by the Brazilian Ministry of Education, grant no 01723/93-8/CAPES.

‡ E-mail: {edjardm, dr, smaill}@aisb.ed.ac.uk

0747-7171/96/110665 + 34 $25.00/0

standardization in the combination process. This is critical when the models were conceived for different levels of time granularity to simulate processes working at different levels of abstraction, but which are somehow related.

In this paper we present a temporal logical reasoning framework to deal with time granularity based on an ontology of time called a *Linear-Cyclic Hierarchy*. Granularity of time is defined by means of a hierarchy of modular sets and an easy mechanism for specifying processes working at different time scales and which can also be cyclical is provided. The logic was used in the representation of seasonal cycles in agroforestry problems, and we show how it can be used as a programming language to develop simulation models working at different levels of time granularities, particularly for ecosystems.

2. Motivation

Granularity is very important if we intend to look at the world at different levels of abstraction, when switching from one level to another may be necessary for the comprehension of the phenomena being observed (Hobbs, 1985). In particular we are concerned with the representation of processes, cyclical or not, working at different time scales. A logical framework for representing and reasoning about this kind of knowledge should deal with the problems of:

(i) Definition of propositions at appropriate levels of temporal granularity.
(ii) Relationships between propositions defined over different time granularities.
(iii) Alignment of temporal domains, allowing events at different temporal granularities to be synchronized.
(iv) Dealing with the "next" temporal operator. For instance, "next time I will play football" may have different interpretations depending on the level of time we are talking about.

Along with these problems we address the need for dealing with cyclical events or processes, at different levels of time and which may interact. The following example shows how such a need arises in an ecosystem model, the agents involved, and the effect of their (possibly cyclical) actions.

> *Example 1*: A piece of forest is composed of 10 trees, each one allocated in a square with sides of 3 m, where the shape of the tree is assumed to be unimportant. Each tree has a growth rate r_i, per some unit of time, which varies according to the season and the level of nutrients in the soil. The growing process of one tree may affect the growth of its neighbours because of the competition for nutrients and light, assuming constant absorption of light per unit area of the canopy of the tree.

The problem of interest in this scenario is to predict the change of height of the trees in a weekly scale of time, and as a consequence their biomass and the biomass of the whole forest. A sophisticated simulation model for this problem should consider as a relevant part of the scenario the following processes.

(i) The rate of water uptake, based on the features of the soil where the tree is placed. This rate would be at some scale of time. Note that the water in-flow of the soil would also change according to the season of the year. So we may need to represent it as a cyclical process.

(ii) The rate of the absorption of nutrients (or uptake), e.g. carbon (photosynthesis) and nitrogen (roots). Another time scale may be needed for this.

(iii) The influence of the concentration of water, nutrients, etc, in the soil, according to the age of the forest. This will be responsible for stopping the increase of tree biomass. Another time scale may be needed.

(iv) Competition for light, assuming the absorption of light per area of the canopy of the trees, and also its height. Another scale may be needed.

(v) External events which change the environment

- natural events such as fire, storms, epidemics of insects, new trees appearing due to natural production of seeds, etc
- cutting some trees down, reforesting some areas, etc.

Let us consider the process involving the leaves and the roots of a tree, and make an intuitive analysis of their effect on the growth rate. The effect of photosynthesis changes in a scale of minutes, since the incidence of light changes minute by minute within a day. On the other hand, the effects of water uptake and nitrogen absorption by the roots are more effectively described on a hourly time scale. The processes clearly work at different time granularities and we need to integrate them to represent their influence in the growth rate of a tree properly. Due to the great complexity of the processes involved, what is usually done, in practice, is either to keep individual models separately (maybe some parameter values or data sets are shared on an ad hoc basis), or one very large imperative model is built. In the latter case, the control structure of all models must be adapted.

The work presented in this paper proposes a temporal logical reasoning framework for problems of this nature.

3. NatureTime Logic Definition

In this section we will present the hierarchical theory of time originally proposed in Mota (1994), which was an attempt to provide a logic-based language to represent concepts of time, following closely the forms of expression used informally in descriptions of ecological systems. The basic idea of our logic is to separate the task of defining relations within the model from the task of computing when these relations hold. We view temporal reasoning as a problem of unification between temporal labels, where the flow of control of program execution is influenced by this. Although this can be seen as similar to unification between sorts in order sorted logics, the presentation of our time theory is not directly based on that approach. We leave this for future work, and any use of the term *sort* in this work does not assume any underlying sorted logic theory.

3.1. BASIC ASSUMPTIONS

Our main concern is to provide a logical framework for which the translation of temporal knowledge from a complex domain (in our case ecological modelling), is not a too painful process. Because the different levels of time are usually expressed by using a label for each one, such references about scales should be part of the model. For this reason, we decided not to "disturb" accepted ways of representing simulation models using computational logic, e.g. Robertson *et al.* (1991). That is, it would be ideal if the temporal

aspects of the program could be thought as labels for components of Prolog-like clause programs, so that we could distinguish the task of defining relations within the model from the task of saying when these relations hold. By taking domain examples in the form of English text (Sinclair *et al.*, 1993; Haggith *et al.*, 1992), we classified the *sorts* of expressions that are normally used to refer about inherently cyclical temporal classes, and that are hierarchically related. For instance,

(i) "the tree grows faster during the rainy season (say February) than in any other season" (from the example in Section 2)
(ii) "The effect of photosynthesis changes on a scale of minutes. On the other hand, the effects of water up take and nitrogen absorption by the roots are more effectively described on a hourly time scale. Both processes influence the growth rate of a tree".

We can see from the first example that it is necessary to talk about things which happen intermittently through the flow of time. From the second example we have processes which can also change at different temporal scales. The main issue in these examples is to merge specifications at different time scales coherently in such a way that the responses of lower levels can influence the responses of higher levels. There may exist some cases in which the opposite effect occurs. For instance, some ecological models consider a forest as "a big leaf", and the behaviour of the forest may affect the behaviour of each individual tree over very long periods of time. Another example from a very different field is inflation, in economics. It is a property of the overall economy, but which affects individuals in different ways. Thus, what is needed is a framework of time which allows us to define cyclical relations and relations at many levels of time granularity.

As an initial step we took actual phenomena in nature which led us to use references of time as in these simple examples. This is depicted in Figure 1, where we view the granularity of time as being a hierarchy of modular sets which are related by a kind of "inclusion relation". At all levels, except one (possibly the highest being considered), time is closed, i.e. time moments are isomorphic to points on a circle (Poidevin and MacBeath, 1995). According to this view of time we can understand, for instance, years as being modular cycles of months, [lunar] months of days, and so on.

The basic mathematical framework to model concepts like "seasons", or cyclical processes and such a *hierarchy of cycles* is modular arithmetic [also called clock arithmetic (Biggs, 1987)]. In this way, each cyclical temporal class is defined by one modular set, and so we permit the succession of time in a cyclical way, where the last element is followed by the first. The theory is based on the following assumptions.

1. *Temporal entity* (TE)—is a reference to a measure of time, e.g. June 24th, 1980. This allows us to define the "previous" and "next" operators without ambiguity, since they will refer to temporal entities at the same level of time granularity. For instance, "next month" would refer to another month, as "previous year" refers to another year.
2. TEs are grouped and circularly ordered, forming *modular temporal classes* (MTC), e.g. December and January are TEs of the class "month", and the last element, December, is followed by the first, January.
3. Temporal classes are modularly sub-divided in other temporal classes by what we call *modular values*. For instance, 60 is the modular value which sub-divides an

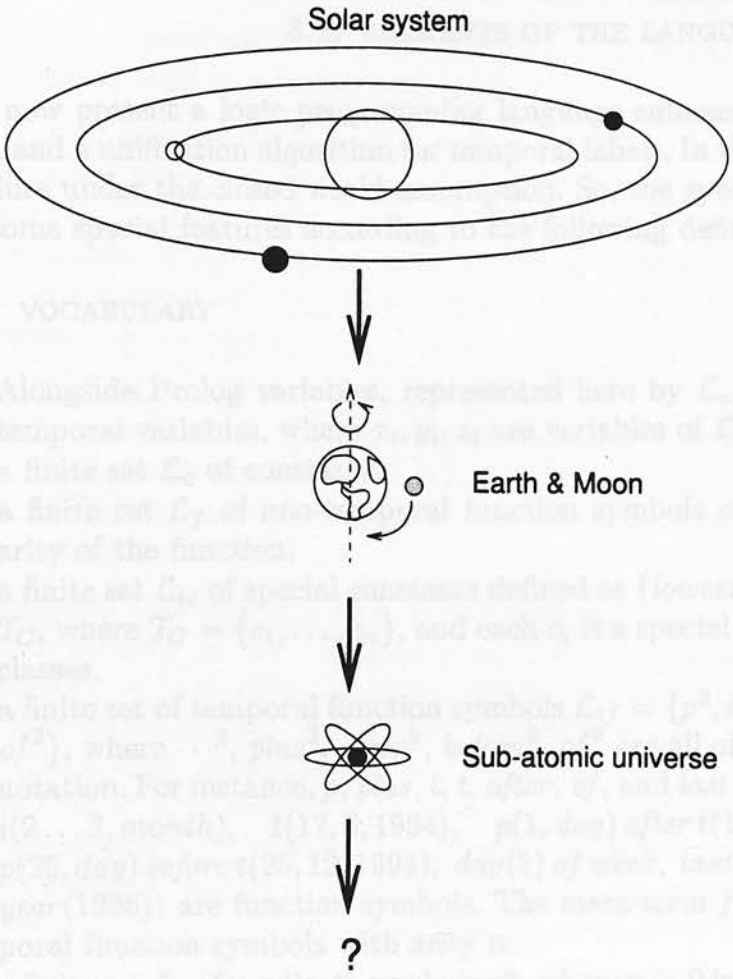

Figure 1. View of the natural events we use to refer about time. Years are cycles of months, months of days, and so on.

hour into minutes. This provides us with a facility to subdivide temporal classes in the way we need.

4. The specification of one MTC defines one level of a time hierarchy. There may exist as many levels of hierarchy as we want. For instance, one could be interested just in days and hours, and so there would be two levels where the second (day) would be defined by the first (hour). This allows us to have multiply nested levels of time granularity.

5. The number of levels of a time hierarchy must be finite. The highest can be considered as the larger interval, and the lowest as the smallest. This is to make the theory a tractable one, because without this restriction there would be no way to compute the operations over our time expressions

6. The highest level of the hierarchy is not circularly grouped to form a MTC, but it is linearly ordered in an infinite sequence. Thus, for each instance of the highest temporal class there is one positive integer. In the example of the previous item, day would be a temporal class with its instances linearly ordered, but not those for minutes which would be circularly ordered. This is to prevent the flow of time being an eternal cycle.

3.2. ELEMENTS OF THE LANGUAGE

We now present a logic programming language enhanced with some special symbols, terms, and a unification algorithm for temporal labels. In this work we also allow negation by failure under the closed world assumption. So, the syntax is basically Prolog-like but with some special features according to the following definition.

3.2.1. VOCABULARY

(i) Alongside Prolog variables, represented here by $\mathcal{L}_v$, there is a disjoint set $\mathcal{L}_{tv}$ of temporal variables, where x_i, y_i, z_i are variables of $\mathcal{L}_v$, s_i, t_i, u_i are variables of $\mathcal{L}_{tv}$.

(ii) a finite set $\mathcal{L}_c$ of constants.

(iii) a finite set $\mathcal{L}_f$ of non-temporal function symbols of the form f^n, where n is the arity of the function.

(iv) a finite set $\mathcal{L}_{tc}$ of special constants defined as $\{lowest, flowtime, infinity, smallest\} \cup \mathcal{T}_C$, where $\mathcal{T}_C = \{c_1, \ldots, c_n\}$, and each c_i is a special constant or names of temporal classes.

(v) a finite set of temporal function symbols $\mathcal{L}_{tf} = \{p^2, i^2, t^n \ldots^2, plus^2, after^2, before^2, of^2\}$, where $\ldots^2$, $plus^2$, $after^2$, $before^2$, of^2 are all of arity 2 but written using infix notation. For instance, p, *plus*, i, t, *after*, *of*, and *last* in $p(2, hour)$ *plus* $p(37, minute)$, $i(2 \ldots 3, month)$, $t(17, 6, 1994)$, $p(1, day)$ *after* $t(12, 3, 1995)$, $p(13, month)$ *plus* $p(25, day)$ *before* $t(25, 12, 1994)$, $day(1)$ *of* *week*, $last(day(2)$ *of* *week* *of* $month(2)$ *of* $year(1996))$ are function symbols. The meta-term f_t^n will be used to refer to temporal function symbols with arity n.

(vi) a finite set $\mathcal{L}_p$ of predicate symbols p^n, where $n > 0$ is the arity of p, and $\{subclasses^2$, on^2, $mod\_temp\_class^3$, mod_value^2, mod_value^3, $change\_mod\_value^3\}$ is a special subset of $\mathcal{L}_p$.

(vii) the set $\mathbb{Z}$ of integers is also part of the vocabulary.

(viii) the propositional connectives for negation, conjunction, disjunction, and reverse implication are represented here by $\neg$, $\&$, $\vee$, and $\Longleftarrow$, respectively. The truth value for true is represented by $\top$, and false by $\bot$.

(ix) a temporal connective @.

3.2.2. CLASSES OF EXPRESSIONS

By using these symbols we define the following classes of expressions, where the capital letters A, B, C are used for formulae.

(i) a *logical term* is defined as usual. Examples of logical terms are *maize*, *grass*, $height(tree(t_1))$, $biomass(forest(f_1))$.

(ii) a *temporal term* (TT) is

- a temporal variable $s \in \mathcal{L}_{tv}$.
- a temporal constant $c_t \in \mathcal{T}_C$.
- a temporal function symbol $f_t \in \mathcal{L}_{ft}$ in one of the following forms.
  - a *period*, which is recursively defined as
    - a *single period* $p(s, m)$ where $s \in \mathbb{Z}^+$ and $m \in \mathcal{T}_C$.

  - a *composite period* P *plus* P', where P is *single period*, and P' is a *period* term at a higher scale than P.
* a *cyclical interval* $i(s \ldots t, m)$, where $s, t \in \mathbb{Z}^+$ and $m \in \mathcal{T}_C$
* a *smallest temporal entity* $t(t_1, \ldots, t_k)$, where for n as the number of elements in $\mathcal{T}_C$, then $k \leq n$, each $t_i \in \mathbb{Z}^+$, and each i corresponds to exactly one element of $\mathcal{T}_C$. This can also be seen as a moment of time.
* a *linear interval* $s_1 \ldots s_2$, where s_1 and s_2 are in the form $t(t_1, \ldots, t_k)$.
* a *collection interval*
  - α where $\alpha \in \mathcal{T}_C$.
  - $f_t(n)$, where $f_t \in \mathcal{L}_{tf}$, and $n \in \mathbb{Z}^+$, and f_t is some $\alpha \in T_C$. For instance, the function symbol of *week*(1) corresponds to *week* of $\mathcal{T}_C$
  - $\alpha(n)$ *of* S, where $f_t \in \mathcal{L}_{tf}$ (same as previous item), and $n \in \mathbb{Z}^+$, and S is a *collection interval.*

A *ground temporal term* is a TT with no variables, e.g. $i(2 \ldots 2, month)$ is a GTT while $i(2 \ldots x, month)$ is not. Examples of TT are $p(3, month)$ *plus* $p(2, day)$, $i(10 \ldots 3, month)$, $t(15, 2, 1994), \ldots, t(3, 4, 1994)$, $minute(43)$, $day(5)$, $week(3)$, $year(12)$, $minute(43)$ *of* day, $minute(43)$ *of* $day(3)$ *of* $week$, $day(1)$ *of month of* $year(1994)$. Note, that *collection interval* is more general than *cyclical interval* in the sense that it may define cyclical and non-cyclical intervals. For instance, $day(2)$ *of week of month* is intended to represent all Mondays of all months, and so is cyclical. However, $day(2)$ *of week of month*(2) *of* $year(1996)$ is not cyclical because it represents the finite sequence of days 5th, 12th, 19th, and 26th of February 1996.

(i) a *first* and *last* TT of a *collection interval* are represented by $first(X)$ and $last(X)$, respectively, where X is a *collection interval*. For instance, suppose X is $day(2)$ *of week of month*(2) *of* $year(1996)$), then $first(X)$ corresponds to exactly the temporal entity $t(5, 2, 1996)$, while $last(X)$ corresponds to $t(26, 2, 1996)$.

(ii) a pure temporal expression (PTE)

- s, if s is a TT, but not a *period term.*
- p *after* s, where p is a *period* term, and and s is a pure temporal expression.

Examples of PTE are $p(3, month)$ *after* $i(10 \ldots 11, month)$, $p(24, year)$ *after* $t(17, 7, 1970)$, $hour(4)$ *of* $day(1)$ *of* $week(1)$ *of* $month(3)$. We say that one of the TT are *canonical forms* of PTE.

(iii) an atomic formula (AF) is

- $\top$ and $\bot$ are atomic formulae
- if $x_1, \ldots, x_n$ are logical terms and $p^n \in \mathcal{L}_p$, then $p(x_1, \ldots, x_n)$ is an atomic formula
- if A is an AF, so is $\neg A$

(iv) classical atomic formulae can be annotated with a PTE by using the temporal operators "@", which means that a classical logical formula is true throughout the whole interval, i.e.

- A, if A is an AF, then it is an atomic temporal formula (ATF).
- $A @ T$, where A is an AF and T is a PTE, is an ATF

Some examples of ATFs are,

- *time_between*$(i(2\ldots4, month), p(3, month))$ which means that *the period of time between February and April, including both, is always equal to 3 months*
- *harvested*$(maize, highlands)\,@\,i(12\ldots1, month)$ which means that *maize is harvested in the high lands throughout the whole interval from December up to January.*

(v) *body* is in one of the forms $A \,\&\, B$, $A \vee B$, or C, where A and B are bodies and C is an ATF. A typical example of a body is *time_between*$(i(2\ldots4, month), p(3, month)) \,\&$ *harvested*$(maize, high\_lands)\,@\,i(12\ldots1, month)$.
When representing different propositions A and B which are true at the same time interval T, it will be required to write $(A \,\&\, B)\,@\,T$ rather than $A\,@\,T \,\&\, B\,@\,T$.

(vi) a well formed temporal formula (WFTF) is

- if A is a positive ATF (non-negated ATF), then A is a WFTF with an *empty body*
- if A is a positive ATF and $B \neq \top, \bot$, and B is a body then $A \Longleftarrow B$ is a WFTF, and A is called the *head*

Note that a WFTF where its *head* is an AF, and *body* is formed only by atomic formulae corresponds exactly to a Prolog clause with no temporal contents.
An example of a WFTF is

$$harvested(tomatoes)\,@\,p(6, month)\ after\ T \Longleftarrow planted(tomatoes)\,@\,T.$$

Which means *if tomatoes are planted throughout an interval T, then they are harvested 6 months after T.*

3.3. THE LINEAR-CYCLIC HIERARCHY STRUCTURE

The temporal structure of time, called *Linear-Cyclic Hierarchy*, is a 4-*tuple* $\mathcal{LC}_H =_{def} \langle \mathcal{E}, \prec, \mathcal{T}_h, \overset{t}{\succ} \rangle$, where $\mathcal{E}$ is a non-empty set of temporal entities (or units of time), $\prec$ is a binary relation of precedence over the set of moments of time in $\mathcal{E}$, $\mathcal{T}_h$ is a finite set $\{c_1, \ldots, c_n\}$ of modular temporal classes, and $\overset{t}{\succ}$ is a partial ordering relation over $\mathcal{T}_h$. This relation induces a special set $\mathcal{T}_{mh} \subseteq \mathcal{T}_h$, $\{mc_1, \ldots, mc_k\}$ and $k \leq n$, called the *main time hierarchy (MTH)*, defined by a sequence of relations as follows.

mod_temp_class$(flow\_time, c_k, infinity)$, where $c_k \in \mathcal{T}_C$
mod_temp_class(c_i, c_{i-1}, m_v), where $1 < i \leq n$, $c_i, c_{i-1} \in \mathcal{T}_C$, and $m_v \in \mathbb{Z}^+$ or $m_v = x\ldots y$ and $x, y \in \mathbb{Z}^+$ and $x < y$.
mod_temp_class$(c_1, lowest, smallest)$, where $c_1 \in \mathcal{T}_C$.

Each pair c_i, c_j of $\mathcal{T}_h$ such that $c_j \overset{t}{\succ} c_i$ holds is related by *mod_temp_class*(c_j, c_i, m_v) and m_v is called the *modular value* of the modular set of c_i which defines c_j. Each c_i is called the name of the class. As $\overset{t}{\succ}$ (properties are in Appendix A is a partial ordering relationship, there will be some MTCs of the set $\mathcal{T}_h$ which will not hold such a relation between them. Note that the set $\mathcal{E}$ is a set of temporal entities which are the forms of TT we have presented, i.e. moment of time, cyclical, linear and collection interval. In the final case, these classes are out of the MTH and are said to be disjointed.

There are two things worth noticing. First, *flow_time* is defined as a special MTC which in fact does not belong to the hierarchy, but it is used to say that the class c_n is the highest class, and that the flow of time will be associated with infinite instances of c_n. Second, the fact that the lowest level of the hierarchy is defined in terms of a temporal constant symbol, i.e. *lowest*, allows us to interpret this structure as a hierarchy of discrete intervals. However, such an assumption is not necessary since we may consider the lowest level as belonging to the set of rationals, which would give us a dense model of time.

In the case that a non-regular MTC is defined, i.e. the modular value m_v is a range rather than a single value, then we need to specify the subclasses, the modular value for each one, and any relationship between them and other levels. Such a kind of MTC is useful for solving the problem related with the irregularity of the real calendar. This is done as follows.

(i) $subclasses(c, l)$, where c is a irregular MTC and c is a list of constant symbols representing the names of the subclasses of c.

(ii) $mod\_value(c, m)$, where c must be an element of a list of subclasses, as defined in the previous item, and $m \in \mathbb{Z}^+$.

(iii) $change\_mod\_value(c_i, s_1 \ldots s_2, p(d, c_j))$, where c_i is an irregular MTC which has its modular value ranging between s_1 and s_2 (positive integers), at every d units of time at the level of c_j, and c_j is a MTC defined by C_i.

(iv) $mod\_value(c_i, I_{c_j}, z)$, where c_i is an irregular MTC, I_{c_j} is an instance of the MTC c_j, which is defined by c_i, and z is the modular value of c_i according to the following recursive definition.

- $mod\_value(c_i, \_, z)$ iff there is some $Z \in \mathbb{Z}^+$ such that $mod\_value(c_i, z)$.
- $mod\_value(c_i, I_{c_j}, z)$ iff both $change(c_i, s_1 \ldots s_2, p(d, c_j))$ and $\mathcal{R}(I_{c_j}, d, s_1, s_2, c)$ hold, where $\mathcal{R}$ is a meta-predicate to represent a temporal relation between I_{c_j}, d, s_1, s_2 that will compute z.

For instance, the real calendar can be defined as follows.

$mod\_temp\_class(flow\_time, year, infinity)$.
$mod\_temp\_class(year, month, 12)$.
$mod\_temp\_class(month, day, 28 \ldots 31)$.
$mod\_temp\_class(day, lowest\_level, smallest)$.
$subclasses(month, [january, february, march, april, may, june, july,$
$\quad august, september, october, november, december])$.
$mod\_value(january, 31)$.
all other cases until $mod\_value(december, 31)$.
$change\_mod\_value(february, 28 \ldots 29, p(4, year))$.
$mod\_value(Class, \_, Z) \Longleftarrow$
$\quad mod\_value(Class, Z)$.
$mod\_value(Class, Year, Z) \Longleftarrow$
$\quad change\_mod\_value(Class, Z1 \ldots Z2, p(MF, year))$ &
$\quad Leap\,Year\ is\ Year\ mod\ MF$ &
$\quad (Leap\,Year = 0$ &
$\quad\ Z = Z2$

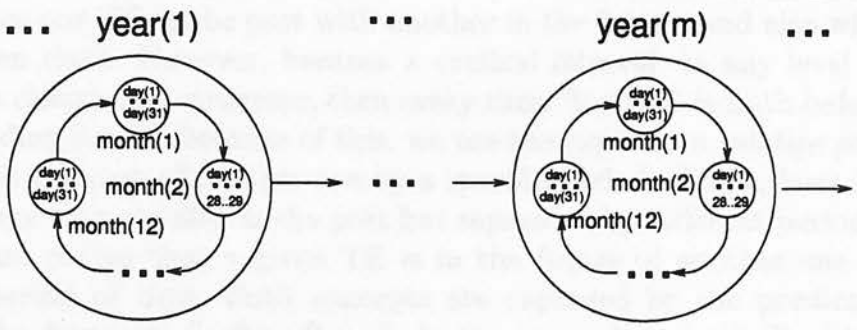

Figure 2. A view of the *Linear-Cyclic Hierarchy* of time. The inner circles represent lower levels of time granularity like *day* and *month*, while the outest circles represent the highest level, like *year*.

$$\vee$$
$$LeapYear > 0 \,\&$$
$$Z = Z1).$$

A view of this hierarchy is depicted in Figure 2.

The temporal entity which we can represent through the MTH is the smallest interval $t(x_1, \ldots, x_n)$. In the calendar defined above the term $t(1, 1, 1996)$ represents the first day of the first month of the year 1996. Along with these MTC we may define other types which do not belong to MTH, e.g. week. This could be defined, for example, as follows.

$mod\_temp\_class(lunar\_month, week, 4).$
$mod\_temp\_class(week, day, 7).$
$mod\_temp\_class(labour\_week, day, 5).$

The temporal entities of this MTCs are represented by collection intervals as described in Section 3.2.2. Although the logic has expressive power to represent such a kind of time interval, we will not explore it in this work since simulation models usually do not need such a kind of temporal reference.

3.4. PAST AND FUTURE RELATIONS

There are two relations for the notions of future and past. The first is for linear intervals, and it is used to implement the usual relations between linear intervals (Allen, 1983; Allen and Hayes, 1985). As linear intervals are represented by a pair of smallest temporal entities (moments of time), which is a structure relating instances of all MTCs of the MTH, then instead of using the less than ($<$) relation directly, as in the case of a single time scale, we have the following linear precedence relation.

DEFINITION 3.1. *Let $R = t(x_1, \ldots, x_k)$ and $S = t(y_1, \ldots, y_k)$ be two smallest intervals. We say that R is linearly precedent to S, written as $R \prec S$ if, and only if*

$x_k < y_k$, *or*
$x_i < y_i$ *and for all* x_j, y_j *such that* $i < j \leq k$ *and* $x_j \leq y_j$.

We also say S is a moment in the future in relation to R.

The second notion relates one TE in the past with another in the future, and also with e period of time between them. However, because a cyclical interval, at any level of anularity, is defined as a closed time structure, then every time "instant" is both before d after any other (including itself). Because of this, we use the expression *relative past* mean that a given TE is the past of another one by a specific period of time, because ere may exist many others that are also in the past but separated by different periods. alogously, *relative future* means that a given TE is in the future of another one in ation to one specific period of time. Both concepts are captured by the predicate ture(S, P, T) to mean the future of S after P is T. In the case where both S and T smallest intervals the definition is trivial and makes use of the temporal precedence ation as defined above and the period of time between them. The definition of this ncept for cyclical intervals is defined as follows, where $\oplus$ is the modular sum operation modular sets (Biggs, 1987).

FINITION 3.2. *Let S, T be two cyclical intervals of a MTC c_i, and suppose there is a TC defined as mod_temp_class(c_{i+1}, c_i, m), and P is a period of time of the level i. We that S is the* relative past *of T, where the period of time between them is P, written future(S, P, T) iff $S \oplus P = T$. We say that T is the relative future of S.*

Now, we use these relations in order to map complex PTEs to canonical forms. This done by using the operations of *up-wave* modular sum and subtraction defined in ppendix B.

FINITION 3.3. *Let $P = p(\Delta, c_i)$ be a period of time, T a temporal interval. Then P er T is the temporal entity in the future of T, defined as*

- $i(s_1 \oplus \Delta \ldots s_2 \oplus \Delta, c_i)$, *if T is a cyclical interval $i(s_1 \ldots s_2, c_i)$*
- $t(s'_1, \ldots, s'_n)\,\omega_{\oplus}\,P \ldots t(t_1, \ldots, t'_n)\,\omega_{\oplus}\,P$, *if T is a linear interval $t(s_1, \ldots, s_n) \ldots t(t_1, \ldots, t_n)$.*

The converse of this operator, *before* is easily defined if we change the *up-wave* modular sum by subtraction. Examples of how the *after* operator works are: $p(3, month)$ *after* $...4, month)$ is equivalent to $i(6 \ldots 7, month)$. $p(2, month)$ *after* $t(1, 10, 1990) \ldots$ $2, 11, 1990)$ is equivalent to the interval $t(1, 12, 1990) \ldots t(12, 1, 1991)$.

3.5. DESIRED INFERENCES

Now, we are going to elaborate on the kind of temporal inferences that should be drawn hen using **NatureTime** for dealing with cyclical events or for the specification of agents having at different levels of time granularity. In the first case, the mechanism is simple. the second case, we will elaborate on a particular type of agent specification that, to knowledge, has not been treated so far. This is for the specification of the behaviour an agent is simulation models, specially those used for ecosystems. We will discuss by an executable temporal (logical) reasoning system is suitable for such a problem, d then what are the problems we may come across and how we can overcome them ing **NatureTime**. Both cases may also be combined as will be shown in Section 5. In ther case, the way in which a temporal sentence will being provable from a knowledge se will differ only in the way the proof is constructed.

3.5.1. TEMPORAL UNIFICATION

Before we describe the desired inferences we need to define the concept of temporal unification which is the core of our logic. Unification as traditionally understood is the process of determining whether two expressions can be made identical by performing appropriate substitutions for their variables.

Temporal unification is the process of determining whether two temporal entities (intervals or moments of time) have some temporal entity in common. In traditional temporal reasoning jargon this is equivalent to finding if they are equal, overlap or one is included into the other (i.e. during, starting or finishing). Note that we do not mention substitution of variables, it is implicit that if one (or both) of the temporal entities is (are) a variable, then one may substitute the other. A full description of how this unification works is given in the next section.

3.5.2. TEMPORAL QUERY AND PROVABILITY

We now give the notion of what one would expect as a correct answer for a given temporal query about a knowledge base. Formally, we have.

DEFINITION 3.4. *Let Δ be a set of temporal formulae (i.e. a knowledge base). For any query $\phi @ \tau$ we want to know if there is an interval τ' (possibly more than one), where τ' occurs within τ (possibly equal to it), such that $\phi @ \tau'$ holds.*

The concept underlying the process of proving a given query is *temporal provability* defined as follows.

DEFINITION 3.5. *Let Δ be a set of temporal formulae, $\phi @ \tau$ a temporal formula. We say that $\phi @ \tau$ is* temporally provable *(or* temporally derivable *from Δ, written $\Delta \vdash_t \phi @ \tau$ if by systematically applying modus ponens, along with standard and temporal substitution, to the set of temporal assertions and temporal logical axioms of Δ we can find a temporal substitution τ' for τ such that we can derive the formula $\phi @ \tau'$, written $\Delta \vdash_t \phi @ \tau'$.*

Given a certain goal, the interpreter we are going to present in the next section searches systematically for derivations of temporal formulae according to the above definition.

Because it is well known that provability intuitively suggests how logical implication can be automated, then we may say that $\Delta \vdash_t \phi @ \tau$ is equivalent to say that $\Delta \models_t \phi @ \tau$, i.e. $\phi @ \tau$ temporal logically follows from Δ.

3.5.3. REASONING ABOUT CYCLICAL EVENTS

Any inference about any cyclical event E can be obtained by, first, specifying an atomic temporal formula involving the event and the interval of time at which such a repetition happens at a given time granularity. In this case the temporal entity T should be a cyclical interval, e.g. $i(s_1 \ldots s_2, c_i)$. Then we write $E @ i(s_1 \ldots s_2, c_i)$ to mean that E *is true throughout the cyclical interval $s_1 \ldots s_2$ at the level i of granularity defined by the MTC c_i.*

Example 2: Suppose we have the following sentences in a knowledge base

$harvest(corn) @ i(5 \ldots 6, month)$.
$harvest(herb\_tea) @ i(2 \ldots 10, month)$.
$harvest(coffee) @ i(8 \ldots 4, month)$.
$harvest(rice) @ I \Longleftarrow (harvest(tea) \,\&\, harevst(coffee)) @ I$.

The types of inference cyclical reasoning about this KB could be:

?- $harvest(corn) @ i(6 \ldots 9, month)$.
?- $\{harvest(corn) @ i(6 \ldots 6, month)\}$
?- $harvest(corn) @ i(3 \ldots 5, month)$.
?- $\{harvest(corn) @ i(5 \ldots 5, month)\}$
?- $harvest(rice) @ T$.
?- $\{harvest(rice) @ i(8 \ldots 10, month), harvest(rice) @ i(2 \ldots 4, month)\}$
?- $harvest(corn) @ t(x_1, x_2, x_3) \ldots t(y_1, y_2, y_3)$
?- $harvest(corn) @ t(1, 3, x_3) \ldots t(31, 5, y_3)$

Note that the last query is more specific in the levels of time required. This will be very useful when we have processes involved at different granularities. The process of reasoning is standard backward chaining. As shown in Mota (1994), it is suitable for representing entailed events (cyclical or not) of temporal knowledge for ecological domain and for reasoning about when an implication holds in a given knowledge base, i.e. for a knowledge base Δ we may be interested to know if $\Delta \vdash P \rightarrow Q$, where P and Q are atomic sentences. However, because backward chaining involves looking back in time for clauses or events, under certain circumstances, forward chaining can be computationally more attractive. One of these is the case of reasoning about simulation models of ecosystems (i.e. simulating the behaviour of some agent), where backward chaining does not seem to provide reliable solutions for testing more complicated problems as in the case of many agents interacting at different scales of time. Our next step will be to show why we also need forward chaining, and for what kind of inferences it is useful.

5.4. REASONING ABOUT AN AGENT'S BEHAVIOUR

There are some cases in which the behaviour of certain entities could be represented by means of differential equations, which is an expressive way of representing the *continuity* of their behaviour. But a continuous representation of time is not always the best for the domain of ecosystems. For example, if we want to represent the behaviour of agents which immigrate and emigrate from one population to another, then it is inconvenient to represent this using continuous functions.

A more usual way of modelling the changes in the state of such agents is to perform them at the time step of their corresponding granularity (sometimes with a fine grained "internal" time scale to approximate continuous change). This yields a discrete approximation to continuous models. However, a discrete approach does not usually allow us to compute the value of some attribute at a time in between two consecutive time steps. For instance, if we have the specification of some ecological entity working at a weekly scale of time, then because its attribute would be updated only at every week we could not obtain its value separately within a week. To overcome this, an additional mechanism seems to be needed beyond the standard way of proving a sentence.

One way of obtaining such a value is to assume that the attribute changes in a linear

fashion, and then using a linear equation to estimate it. For more complex processes a non-linear behaviour might be assumed and the value would be computed in the same way. When doing this, we have to assume that the process does not interact with other processes (possibly of other agents) that may affect the attributes in question. The reason for such an assumption is that the function used to estimate the desired value needs the values of the attribute at the end points of the interval between the consecutive time steps, i.e. the previous and the future values. However, if there is such an interaction, then we cannot estimate the future value because it may happen that the other process(es) have influenced so much that the effect over the attribute can be much more than the expected. Moreover, to estimate such a future value it is needed to know the wanted value which leads us to a situation of "deadlock".

For simplicity we avoid interpolation between time points. In this case every entity with its behaviour specified at a coarser level will have its state changed only at that level, i.e. any query about its state at any time between two consecutive time steps will always give the computation for the last one. Based on these assumptions, the temporal reasoning system should reason about queries like:

1. What is the value of an attribute of an entity at any given time?
2. When will any two entities (no relation between them) have the same value?
3. What is the value of a given entity at a given time which interacts with another entity which works at different time scale?

There are two important components to this problem. First we want the deduction process to be able to search for a value which satisfies one query, but this value should be determined by a given temporal entity. This time can be synchronous or asynchronous in relation to the updating time steps of the attributes of the ecological entity we want to know something about. Second, the computation of the value of an attribute of an agent at a given time i depends on the value at time $i-1$. Because of this, a simulation model can be interpreted as being a relation between the state of the agent in the past and its state in the future ruled by certain conditions of state transition.

This idea of constructing the future state based on the past is known as *declarative past and imperative future* (Gabbay, 1989). In this view programs should be expressed in a temporal language which could be read in a declarative way, and when the program runs it should construct a model for the temporal sentence it is intended to represent (Gabbay and Reynolds, 1995). Relating it to the usual simulation models of ecosystems are developed, this is equivalent to describing the behaviour of agents using a temporal logic. The simulation of the behaviour of the agent through the flow of time corresponds to the construction of a model, or running the program for a temporal sentence at each time step. As our language deals with explicit representation of time, the usual way of representing *past* $\rightarrow$ *future* in this work should consider the length or period of time between past and future (or present). As far as simulation models are concerned, such an imperative formulation can be represented in **NatureTime** as in the following schematic way.

$$A' \,@\, P \; after \; T_p \Longleftarrow A \,@\, T_p \,\&\, \mathcal{R}(A, A').$$

where A represents the information about an ecological species, T_p the previous time, P is normally in the form $p(1, C)$, C is an MTC, and the second order predicate $\mathcal{R}$ is intended to represent the sequence of formulae which involves A and A' to produce their

lationship within the flow of time. However, Gabbay's approach proposes not to use he traditional execution mechanisms based upon resolution and refutation. In this paper e do not need to make such an assumption. If A and A' both simply refer to the same redicate (e.g. A is $value(a, X_p)$ and A' is $value(a, X_f)$), then we do not have to worry bout branches in the search space. We simply follow a single chain of conclusions until e have reached some desired (possibly the final) time.

For instance, suppose we have the following hypothetical agent specification.

. $value(a, 100) @ t(1, 1, 1)$.

. $value(a, X_f) @ p(1, week)$ *after* T_p

$\Longleftarrow$

$value(a, X_p) @ T_p$ &

X_f *is* $X_p + 1$.

The execution of this program would generate the value of a for every week, but without ing the traditional combination of backtracking and recursion until the initial state. stead, the generation of the model for the sentence should get the previous state of the mputation (past) in order to produce the next state (future). For this example we have following table with the first steps of execution for a goal $value(a, X) @ T$, showing present state of computation, the sentence being matched for each backtracking step the generation of the future state. After the initial state the sentence used is always second, and to compute a new state it is not necessary to reach the initial state again backward chaining would do.

Present	Sentence matched	Future
—	1	$value(a, 100) @ t(1, 1, 1)$
$value(a, 100) @ t(1, 1, 1)$	2	$value(a, 101) @ t(1, 7, 1)$
$value(a, 101) @ t(1, 7, 1)$	2	$value(a, 102) @ t(1, 14, 1)$
$value(a, 102) @ t(1, 14, 1)$	2	$value(a, 103) @ t(1, 21, 1)$

4. The Enhanced Meta-interpreter for Temporal Reasoning

n this section we will show the specification of the simple meta-interpreter of the **tureTime** system. We will show the clauses of the extended temporal meta-interpreter, h an explanation for each extension.

4.1. A META-INTERPRETER FOR NATURETIME

The meta-interpreter is an extension of the one presented by Sterling and Shapiro 86). Because we are interested in problems where the specification of the behaviour agents through the flow of time is basically a clause where the state of the agent in past is related to the (present or) future, then we also allow the meta-interpreter to l efficiently with this kind of clause. This will be presented in Section 4.1.2.

The extensions are basically twofold. First, the predicate *clause* must also identify well-formed temporal formula as specified in Section 3.2.2; second, the introduction of a special unification for PTEs. Note that most of the unification will still be done by standard Prolog unification, except when involving PTEs. However, as we treat temporal reasoning as a problem of unifying PTEs, then we need to control the part of the unification which deals with it. Because of this, the standard $solve^1$ predicate (Sterling and Shapiro, 1986) is changed to be a binary relation $solve^2$. The meta-interpreter accepts queries which are temporal formulae. It succeeds if the query holds throughout some interval within the given PTE as specified in Definition 3.6. So, the first argument is the temporal formula to be solved, and the second the result of the computation for the first argument be true. In what follows, $system(X)$ succeeds if X is a system predicate but not a negation. The meta-interpreter is defined as follows.

1 - $solve(X, X) : -$
$\quad \neg temp\_formula(X),$
$\quad (clause(X \Longleftarrow Y),$
$\quad\ solve(Y, \_)$
$\quad\ ;$
$\quad\ system(X),$
$\quad\ call(X)).$

2 - $solve(\neg X, \_) : -$
$\quad \neg solve(X, \_).$

3 - $solve(A @ T1, A @ T3) : -$
$\quad \neg A = (\_ \,\&\, \_),$
$\quad clause(A @ T2),$
$\quad temp\_unify(T1, T2, T3).$

4 - $solve(A @ T, A @ RT) : -$
$\quad A = (A1 \,\&\, A2),$
$\quad solve(A1 @ \_, A1 @ T1),$
$\quad solve(A2 @ \_, A2 @ T2),$
$\quad temp\_unify(T1, T2, RT1),$
$\quad temp\_unify(T, RT1, RT).$

5 - $solve(A \,\&\, B, A1 \,\&\, B1) : -$
$\quad solve(A, A1),$
$\quad solve(B, B1).$

6 - $solve(A \vee B, R) : -$
$\quad (solve(A, R)\ ;$
$\quad\ solve(B, R)).$

7 - $solve(A @ T1, A @ T3) : -$
$\quad clause(A @ T2 \Longleftarrow Body),$
$\quad solve(Body, \_),$
$\quad temp\_unify(T1, T2, T3).$

The first clause deals with classical formula in the usual way. The second implements the negation by failure as using the standard way of negating if it cannot be proved. The third clause consults the Knowledge Base (KB) to check whether we can directly

ich $A@T1$ to some unit clause or not. The fourth clause deals with composite events, ere we want to know if there is some time interval throughout which they can happen together. Clauses 5 and 6 are just another way of re-writing the standard clauses logical conjunction and disjunction, and so do not need any detailed explanation. last clause is the clause for logical implication, which gives an alternative for the two clauses if they fail. To understand the clauses of the meta-interpreter we need nderstand the declarative interpretation of each one; they should be interpreted as ows.

- *A is true throughout T3, if A is not a composite event, and A is true throughout T2, and T3 is the temporal unification of T1 and T2.*
- *A is true throughout T if, A is a composite event consisting of A1 & A2, and A1 is true throughout T1, and A2 is true throughout T2, and RT1 is the temporal unification of T1 and T2, and RT is the temporal unification of T and RT1.*
- *A is true throughout T3 if, the A@T2 is the head of a temporal Horn clause with body Precondition, and the body can be solved, and T3 is the temporal unification of T1 and T2.*

the first (1) and in the last clause (7), when the Y and *Body* are solved, any rrence of temporal entities within them will be substituted by standard unification. value computed by the second argument is ignored here. This can give unsound lts if there are shared variables present. We restrict programs and queries to avoid this ation. The ideal solution is either if the solver carries a list L of temporal variables and s them only after solving all temporal formulae in a wftf, or having a sort of memory storing operations over PTE and combining those stored with the new one during process of deduction. This is similar to the idea of *environment* (van Emden, 1984) in the implementation of Prolog machines. However, this would require a more isticated meta-interpreter, and for the purposes of this work this simple solution ides a powerful mechanism for a significant subset of temporal problems.

2. DEALING WITH SIMULATION MODELS

s we have discussed in Section 3.5.4 it is interesting to deal with simulation clauses different way. The process of reasoning should be Forward Chaining rather than ward, and we also want to obtain the state of agent at any time in between two ecutive time steps of its behaviour, i.e. asynchronous information. Although the extension defined in the last section can be used for the specification of simple lation models (Mota *et al.*, 1995a), we have shown (Mota *et al.*, 1995b) that for more plicated models, where it is needed to obtain asynchronous information, the *solver*[2] not offer mechanisms for this. In this work we will provide the meta-interpreter such a facility. Because we are considering discrete models, then every entity with ehaviour specified at coarser levels will have its state changed only at that level, i.e. query about its state at any time between two consecutive time steps will always the computation for the last one.

r a given goal $A@T$ the solving process should first check if the time requested is a time. If it is the case, then verify if this formula can match with part of the body of mporal formula like the schemata given in Section 3.5.4, which we call a *simulation* e. After this, instead of using the meta-interpreter as above (which would apply

backward reasoning), the search should re-use, at every time step, the last value for the state of the entity in order to reason forwards.

What if the time in the goal is not a ground temporal term? We will use the same solution, except that the searching process does not need to check the next time with the previous one in order to stop the search.

In the more complicated case, the searching process basically checks if the solution found is that of the specified time. If it is not the case, then if the given time T is the future of T_p and the PTE $P\ after\ T_p$ is not the future of T, a new instance of the simulation clause is used to continue the search. More formally we have.

DEFINITION 4.1. *Given an atomic temporal assertion $A_i @ T_i$ (or the initial state for A), a goal $A_1 @ T$, one (or more mutually exclusive) simulation clause schemata*

$$C = A' @ P\ after\ T_p \Longleftarrow A @ T_p \,\&\, Constraints,$$

where A, A', A_i and A_1 have the same predicate name and arity. Then, the forward computation which will search for a solution for the goal $A_1 @ T$ is given as follows.

if T is ground temporal term then

reduce T to a canonical PTE, say T_1
$T'_i = T_i$
$A'_i = A_i$
while $T'_i \prec T_1$ then

get a new instance C' of C, matching $A @ T_p$ with $A'_i @ T'_i$, and
future(T'_i, P, T'_j) holds and
if $T_1 \prec T'_j$ holds then stop the search, otherwise
if $\neg T_1 \prec T'_j$ holds and Constraints is found to be true by the solver clauses then
$T'_i = T'_j$ *and* $A'_i = A'$

if T is a temporal variable then

get a new instance C' of C, matching $A @ T_p$ with $A'_i @ T'_i$, and
future(T'_i, P, T'_j) holds and
Constraints is found to be true by the solver clauses then
$T'_i = T'_j$ *and* $A'_i = A'$

return $A'_i @ T'_i$

Note that this algorithm is guaranteed to terminate, in the case that T is a ground TT, because of the fact that the condition $T'_i \prec T_1$ will eventually fail, or $T_1 \prec T'_j$ will eventually hold. If more solutions are requested, then backtracking over the clause selection strategy is possible to find another one only if T is not a ground TT because there will always be a T'_j such that *future*(T'_i, P, T'_j) holds. But in the case $T1$ (the reduced form of T) is a ground TT, then once $\neg T'_i \prec T_1$ holds the computation will never get new instances of C, and so no more solution will be found.

4.2. GENERAL UNIFICATION ALGORITHM

The specialized unification algorithm we developed to treat PTEs consists of two steps. First, every complex PTE is reduced to a canonical form of temporal entities. Second, as

onical forms of a temporal entity are in fact intervals of time, or collections of them, n they are unified according to the usual relations between time intervals. These are ed on the future and past relations, and also in the relation between periods of time at erent levels of granularity. The result of a unification between two temporal entities is their most general unifier as is traditional, but rather the canonical form of temporal ity which is the result of their matching, and this can involve more than one temporal ity as an example will show. The *temporal unification* is defined as follows.

FINITION 4.2. *Let t_1, t_2 and t be three PTE. We write temp_unify(t_1, t_2, t) to mean t t_1 and t_2 are unifiable and t is the unified term iff either*

- $t_1 = t_2 = t'$ *and t is the reduction of t', written reduce(t, t'), or*
- $t_1 \neq t_2$, *and*
  - t_1 *is reducible to rt_1 - reduce(t_1, rt_1) and*
  - t_2 *is reducible to rt_2 - reduce(t_2, rt_2) and*
  - t *is reducible to t' - reduce(t, t') and*
  - t' *is the matching of rt_1 and rt_2 - unify_units(rt_1, rt_2, t').*

or instance, suppose we want to unify $i(2 \ldots 10, month)$ (or all intervals from Febru- to October) with $i(8 \ldots 4, month)$ (or all intervals from August to April). In this e there will be two different instances of the canonical form of cyclical interval, i.e. $\ldots 4, month)$ and $i(8 \ldots 10, month)$. How unification works is related to the unification emporal units as defined in Section 4.4.

4.3. REDUCTION OF TEMPORAL TERMS

he reduction algorithm is divided into two groups of clauses. The first consists of clauses to deal with canonical forms of temporal expression. The second, deals with plex forms. First we introduce what we mean by temporal variable: a term t is a poral variable if it is a logical variable, i.e. if $t \in \mathcal{L}_v$, or a canonical form of PTE re all its elements are logical variables. Based on it we have the following reduction rithm.

FINITION 4.3. *Let t_1 be a PTE, and t_2 a canonical PTE. We say that t_1 is reducible , written reduce(t_1, t_2) iff one of the following holds.*

- $t_1 = t_2$, *and t_1 is a temporal variable, temp_var(t_1).*
- $\neg$*temp_var(t_1) and is in the form $t(x_1, \ldots, x_k)$, so $t_2 = t_1$*
- $t_1 = s_1 \ldots s_2$, *and $t_2 = rs_1 \ldots rs_2$ where*
  - s_1 *is reducible to rs_1 and*
  - s_2 *is reducible to rs_2.*
- $t_1 = t_2 = i(s \ldots t, c)$
- $t_1 = t$ *after* Δ *and*
  - t_1 *is reducible to rt_1 and*
  - t_2 *is reducible to rt_2 and*
  - *future(rt_1, Δ, rt_2) holds.*

- $t_1 = t$ *before* Δ *and*
  t_1 *is reducible to* rt_1 *and*
  t_2 *is reducible to* rt_2 *and*
  future(rt_2, Δ, rt_1) *holds.*

For instance, $p(1, week)$ *after* $(p(1, week)$ *plus* $p(3, day)$ *after* $t(14, 2, 1994))$ is reduced to $t(1, 3, 1994)$.

4.4. UNIFICATION OF TIME UNITS

The unification on the level of units is just the matching of temporal entities. This has to deal with linear and cyclical intervals, and also intervals of both types. The linear unification is based on the $\prec$ ordering relationship of the bounding temporal entities of the interval, and the modular unification is based on the matching of circular intervals. In both cases we use the relations *during* and *overlap* (Allen and Hayes, 1985) taking into account that the "last" element of a modular set is followed by the first. Because of this, the linear match has four different cases, and the modular case six as described in Appendix C.

Its definition is as follows.

DEFINITION 4.4. *Let* s_1, s_2 *and* s_3 *be three canonical forms of PTEs. We say that* s_3 *is the unit matching from* s_1 *and* s_2, *written* $match\_units(s_1, s_2, s_3)$ *if one of the following holds.*

- $s_1 = s_2 = s_3$.
- s_1 *and* s_2 *are linear intervals, and* s_2 *is during* s_1, $s_3 = s_2$.
- s_1 *and* s_2 *are cyclical intervals of a MTC* c, *and*
  $mod\_temp\_class(c', c, mv)$ *holds and*
  s_3 *is the cyclical interval from the modular matching between* s_1 *and* s_2. *(could be more than one matching)*
- s_1 *is a cyclical interval and* s_2 *is a linear interval and*
  s_1' *is a* linear instance *of* s_1, *written* $time\_instance(s_1, s_1')$, *and*
  s_3 *is the linear matching between* s_1' *and* s_2
- s_1 *is a linear interval and* s_2 *is a cyclical interval and* $match\_units(s_2, s_1, s_3)$ *holds.*

Note that *time_instance* creates a linear instance of a cyclical interval. As a cyclical interval represents a collection of linear intervals, there may exist many instances of it in the level of linear intervals. For this reason, this is one of the most important features of the logic because without it, the concept of cyclical interval would be useless. In a pure linear model of time, it is necessary to introduce some "expert" computation over recurrent representation, in order to obtain reasoning about cyclical events. In our work, the use of cyclical interval provides an easy and elegant mechanism to represent and obtain reasoning about cyclical events and processes, since this new type of interval represents many instances in the linear level.

Example 3: Suppose we have the following sentences in a knowledge base

$harvest(corn)\ @\ i(5\ldots6, month)$.
$harvest(herb\_tea)\ @\ i(2\ldots10, month)$.
$harvest(coffee)\ @\ i(8\ldots4, month)$.
$harvest(rice)\ @\ I \Longleftarrow (harvest(tea)\ \&\ harevst(coffee))\ @\ I$.

In what follows, the symbols ": |" and ">>" represent the query and answer prompts, respectively, of the **NatureTime** system. When a query is done the dialog interface calls the *solve*[2], and shows the second argument of it as the answer.

: | $harvest(corn)\ @\ i(6\ldots9, month)$.
>> $harvest(corn)\ @\ i(6\ldots6, month)$
: | $harvest(corn)\ @\ i(3\ldots5, month)$.
>> $harvest(corn)\ @\ i(5\ldots5, month)$
: | *more.*
>> *Sorry, no further solution is possible.*
: | $harvest(rice)\ @\ T$.
>> $harvest(rice)\ @\ i(8\ldots10, month)$
: | *more.*
>> $harvest(rice)\ @\ i(2\ldots4, month)$
: | *more.*
>> *Sorry, no further solution is possible.*

The *more* command allows all possible solutions by backtracking. In the next section e shall show two examples of simulation models specified in our language, and how they e solved.

5. Simulation Models in NatureTime

This section presents the application of our logic language to develop a simulation odel for the growing process of the trees of the example of Section 2, and we also present other example of two temporal entities working at different time scales and which eract with one another. After this, we point out the limitations of this implementation d of the language.

5.1. EXAMPLE OF TREE GROWING PROCESS

Usually, a discrete model of the height of a tree t_i at a given time $t+1$ might be resented by an equation of the form

$$H_i(t+1) = H_i(t) + r_i H_i\left(1 - \frac{H_i}{MAX_{H_i}}\right), \tag{5.1}$$

ere MAX_{H_i} is the maximum height that a tree t_i can reach, r_i is the intrinsic growth e of t_i, and H_i is the height of t_i. Usually, r_i is assumed to be an average value per ne unit of time. However, it does not explicitly model the interaction among trees, ich may affect this rate. The influence on a tree t_i by other trees is approximated as nction of their height and their distance from t_i. This is intended to represent how acquisition of biomass of other trees affects t_i. Basically, the taller a tree t_j, the more ct it will have on the growth rate of t_i, and the further t_j is from t_i the less effect ill cause. This will be represented by the quantity $k\frac{H_j}{d_{ij}}$, where k is a constant which

would normally be determined empirically, and d_{ij} is the distance between t_i and t_j. The increase of the height at every time step will be given as follows.

$$\Delta'_H = \left(r_s - \sum_{j=1}^{nb} k\frac{H_j}{d_{ij}}\right) H_i \left(1 - \frac{H_i}{MAX_{H_i}}\right), \tag{5.2}$$

where r_s is the standard growth rate of a tree if no influence is present, nb is the number of tree neighbours of t_i.

In our previous scenario, as the scale of time used was week, we could define another MTC within the hierarchy defined, that is $mod\_temp\_class(week, day, 7)$. The rainy season, assumed to be only in February, can be written as $season(rain) @ i(2 \ldots 2, month)$. So, in every instance of February, within a MTC of year, it will rain.

The height of a tree can be calculated by using the equation (5.1), but we have to make the growth rate r_i a function of the interaction between the tree and its neighbour trees as in equation (5.2). To do this, we first define a predicate to represent the neighbours of a tree along with the distance between them. The height must be calculated for each time step. This will be represented by the following predicate definition, where the base case is the initial height for each tree.

$height(Tree, H) @ p(1, week)\ after\ T$
$\Longleftarrow$
$height(Tree, H1) @ T\ \&$
$max\_height(Tree, MAX)\ \&$
$real\_gr(Tree, T, \ldots, (p(1, week)\ after\ T, RGr))\ \&$
$neighbours(Tree, NTrees)\ \&$
$influences(NTrees, T, R)\ \&$
$sum(R, TR)\ \&$
$(Gr\ is\ RGr - TR)\ \&$
$(C\ is\ Gr * H1 * (1 - H1/MAX))\ \&$
$(H\ is\ H1 + C).$

The *influences*/3 predicate represents the influence of the neighbours of a tree *since* the time T. The *sum*/2 predicate represent the relation between a list of values and a number which is the sum of these values. The *real_gr*/3 gets the supposed "real" growth rate of the tree throughout the interval of one week. The *real_gr*/3 can be defined in a standard Prolog style, for example, as follows

$real\_gr(Tree, T, Gr)$
$\Longleftarrow$
$season(rain) @ T\ \&$
$growth\_rate(Tree, LowGr)\ \&$
$(Gr\ is\ 1.2 * LowGr).$
$real\_gr(Tree, T, Gr)$
$\Longleftarrow$
$\neg\, season(rain) @ T\ \&$
$growth\_rate(Tree, Gr).$

Gr is the growth rate if T is a rainy season and *Lowgr* is a standard growth rate, then $= LowGr * 1.2$. Gr is the growth rate if it is not true that T is a rainy season. Note t we call the *solve* meta-interpreter to use its facilities of unification as we saw in tion 4. The complete knowledge base of our example is shown in the Appendix D. Below, we show the simulation of the growing process. Note that when the simulation aves" the rainy season the growth rate decreases, as expected.

$height(t1, H)\,@\,T.$
$height(t1, 1)\,@\,t(14, 2, 1994)$
more.
$height(t1, 1.01)\,@\,t(21, 2, 1994)$
more.
$height(t1, 1.02)\,@\,t(28, 2, 1994)$
more.
$height(t1, 1.03)\,@\,t(5, 3, 1994)$
more.
$height(t1, 1.037)\,@\,t(12, 3, 1994)$
more.
$height(t1, 1.045)\,@\,t(19, 3, 1994)$
more.
$height(t1, 1.054)\,@\,t(26, 3, 1994)$
more
$height(t1, 1.062)\,@\,t(3, 4, 1994)$

5.2. TWO ECOLOGICAL SPECIES WORKING AT TWO TIME SCALES

he following example is part of our discussion on the granular aspects of time in ulation models of ecosystems (Mota *et al.*, 1995b).

> *Example 4*: The ecological entities are a tree growing, called simply *tree* and an insect (or a cloud of insects) called *bug*. The *tree* has its growth rate affected by the *bug* only if it flies on the top of the *tree*. The height reached by the *bug* will depend on the height of the *tree*, and the height of the *tree* depends on the time spent by the *bug* at a certain position, say 5 m. For simplicity, instead of using the logistic equation for the growing process, the tree grows 0.5 m every week. The time scale of the bug's movement is considered to be day.

here is an interaction between these two entities, and we will assume that the tree uld get the "progress" of the bug's position in a period of one week. However, in the 's behaviour specification there is no need to explicitly represent the scale of the *bug*, ause it could be another entity interacting with it. Then we shall use the predicate e(*time*, *Object*, *Process*, *MTC*) to specify at which scale of time the process of a given ty works. In this way we have the following facts in our KB.

$th\_rate(tree, 0.5).$
$e(time, bug, movement, day).$
$e(time, tree, growing, week).$
$nd(height, growing).$
$e(height, tree, 9)\,@\,t(1, 1, 1).$

$value(pos, bug, 6) @ t(1, 1, 1).$

Now we need a general way to capture the progress of an attribute of a given agent throughout a certain period of time. We will use the predicate *progress*[4] to represent the progress observed of the value of an attribute *Att* of an agent *Obj*, from a given temporal entity T during a given period of time P, and the progress will return in a list of all the values for *Att* during T. A simple specification for *progress*/4 is in Appendix E, and a more deep discussion on the specification of interacting agents working at different levels of time granularity is out of the scope of this work. More details can be found in Mota *et al.* (1995b). The specification of the *tree*'s growing process can be, for example, as follows.

$value(height, tree, H) @ p(1, week)\ after\ T$
$\Longleftarrow$
$value(height, tree, Hi) @ T\ \&$
$progress(value(pos, bug, \_) @ T, p(1, week), L)\ \&$
$growth\_rate(tree, GR)\ \&$
$influence(GR, L, RealGR)\ \&$
$(H\ is\ Hi + RealGR).$

The specification of the *bug*'s movement can be as follows.

$value(pos, bug, PB) @ p(1, day)\ after\ T$
$\Longleftarrow$
$value(pos, bug, PBi) @ T\ \&$
$value(height, tree, H) @ T\ \&$
$new\_pos(PBi, H, PB).$

The *new_pos*[3] simply implements a change of the bug's position, and it is also in the Appendix E. For this specification we have the following results for the simulation of the bug's position.

$| : value(pos, bug, Pos) @ T.$
$>> value(pos, bug, 6) @ t(1, 1, 1)$
$| : more.$
$>> value(pos, bug, 8) @ t(2, 1, 1)$
...
$| : more.$
$>> value(pos, bug, 8) @ t(8, 1, 1)$

For the tree's growing process we have.

$| : value(height, tree, H) @ T.$
$>> value(height, tree, 9) @ t(1, 1, 1)$
$| : more.$
$>> value(height, tree, 9.44) @ t(8, 1, 1)$

> $value(height, tree, 11.2)\,@\,t(6,2,1)$
: *more.*
> $value(height, tree, 11.64)\,@\,t(13,2,1)$

This just shows the behaviour through the flow of time. In the case of a query about e value of the tree's height at any time we will have the following results, as expected.

: $value(height, tree, H)\,@\,t(10,1,1)$.
> $value(height, tree, 9.44)\,@\,t(10,1,1)$
: $value(height, tree, H)\,@\,t(10,2,1)$.
> $value(height, tree, 11.2)\,@\,t(10,2,1)$

Note that queries were for time values in between two synchronous time steps of the e's growth.

6. Analysis and Related Work

6.1. STRENGTHS OF THE NATURETIME SYSTEM

NatureTime is a comparatively simple logic which does not stray far from the tra- ional style of mainstream logic programming—yet it can deal with a wide range of e problems commonly encountered in ecological modelling and simulation. Moreover, en much of the ecosystem is stable (i.e. unchanging when events happen) then most the updating would be redundant. By using a logic we have presented in this paper only need to deduce changes to the relevant aspects of the ecosystem. Thus if our ulation needs to include things working at a very small temporal granularity (like a le insect) together with things which change slowly (like a tree), it is redundant to date the state of the tree with every event affecting the insect. Our system does not olve this redundancy as we have demonstrated in the Example 3 of Section 5.2.
n terms of representation, **NatureTime** has a simple and elegant mechanism for representation of cyclical knowledge by using the concept of cyclical intervals. This ws us to write $harvested(coffee)\,@\,i(8\ldots4, month)$, without the necessity of saying harvesting can occur every year, since the MTC month is included in the definition ear, and year is flow of time. This can also be used to create another MTC which are nested within the main hierarchy. For instance,

$temp\_class(labour\_week, day, 5)$.
$temp\_class(labour\_month, labour\_week, 4)$
$temp\_class(lunar\_month, week, 4)$.

this paper we did not present any mechanisms to deal with more complex cyclical rvals, and other temporal entities like collection intervals and fluctuation of temporal ties over other temporal entities. This is basically related to introduce mechanisms nference to reason about sentences like "all Mondays of 1996", although it does not

seem to be useful in simulation models. Such a treatment, here, would distract from the main point of this work.

The temporal operator *after* allows a more legible reading of some temporal statements, and it represents well enough the relation between past and future at different levels of time granularity. This leaves the user free to write his/her inference rules without the necessity of using properties and relations between temporal units. It is just needed to understand what a temporal formula is intended to mean and how we can map temporal knowledge to the forms of expressions of the language. For instance, the sentence *grass is free to grow up 1 month* after *a certain month X if sheep use the meadow up to X* can be easily translated to our logic as follows.

$$grow\_free(grass, meadow)\,@\,p(1, month)\ \ after\ i(X \ldots X, month)$$
$$\Longleftarrow$$
$$use(sheep, meadow)\,@\,i(Z \ldots X, month).$$

6.2. LIMITATIONS OF THE NATURETIME SYSTEM

The main limitation of **NatureTime** with earlier approaches using traditional computational logic, is its exponential search requirement in the case we have many agents interacting. However, because it uses a forward chaining strategy to re-use the previous computation of a given attribute, then the reduction of search space is considerable.

Another limitation which does not seem to affect the application for the more precise problems we proposed to tackle, is the lack of a suitable temporal connective for the representation of sentences like "it rained from 3 pm to 5 pm sometime last week". This could be the operator $\diamond$ usually known as "sometimes".

6.3. RELATED WORK

One early investigation on the representation of time clock on any scale was proposed in Ladkin (1986a), where interval calculus (Allen, 1983) is extended to achieve a time framework where different time units (TU) can be specified. This extends the idea of *convex* to *union-of-convex* intervals (Ladkin, 1986b), where there may exist gaps between *convex* intervals. The representation of *Basic Time Units* is a sequence like $[year, month, day, hour, \ldots]$. In Ladkin (1987) it was shown that by introducing appropriate relations between intervals such a sequence gives a suitable representation for a convex rational interval structure. We provide a similar entity that we called *smallest interval* which also has as many elements as desired, but we reach this representation from different concepts, i.e. this will depend on the number of MTCs defined. In **NatureTime** it seems to be a bit easier to define calendars, and we incorporate circularity within the model. Although *union-of-convex* intervals can be compared to our cyclical interval, Ladkin did not explore the subject of dealing with cyclical events.

A similar approach was proposed in Leban *et al.* (1986), where the basic idea is to use a set of primitive collections to specify other collections by using two operators, *slicing* and *dicing*, in order to select intervals from collections of intervals. Each primitive is defined by specifying the intervals of which it is composed. In this approach, circular aspects of time can be obtained from the *δ-values* which are treated as if they were a circular list. Although this approach was shown to be useful for reasoning about scheduling, it does not really deal with different granularities of time because it does not seem to be clear

ow we can we specify relationships between propositions defined over different time ales. Furthermore, the way in which circularity is obtained is very much dependent on e implementation of circular list rather than the concept itself, although it gives a close ea of it.

Another approach for representing cyclical events was proposed by Koomen (1989), hich is based on Allen's system (Allen, 1984; Allen and Hayes, 1985). The idea is define a recurrent event e by stating explicitly that it is true repeatedly over an terval I, i.e. RT(I, e). However, it is not clear whether the interval I is *convex* or not, d it is not obvious how to use the mechanisms of inference to model cyclical events in "natural" way to obtain appropriate inferences. As this approach does not deal with etric information, no representation of propositions at different time scales is possible.

A more pragmatic approach for dealing with time granularity was proposed in Dean 989), in order to speed up the information retrieval on a large temporal data base aintained by the *time map system*. This work proposes a hierarchical framework of me such that events at different levels of abstraction can be easily represented and trieved by using a structure similar to the usual calendar. The hierarchy over a linear ructure of time is obtained by the concept of a *partitioning scheme*, which is a sequence partitions $P_1, P_2, \ldots, P_n$ of the set of reals, in such a way that for each $i < n$ if an terval I belongs to a partition P_i, then there is a set of time intervals in P_{i+1} such that I partitioned by it. Although this approach shows to be very successful in the context data base maintenance, the aspect of representing and reasoning about cyclical events s not explored. Moreover, it does not seem to be a suitable approach for modelling ore complex problem in the real world. As we often want to obtain some prediction inference rules, and not only retrieve assertions about temporal knowledge, then this proach is not suitable for the type of problem we are dealing with.

A more recent approach (Ciapessoni *et al.*, 1993; Montanari, 1994) proposed a many-ted first order logic augmented with temporal operators and a metric on time to deal th time granularity. This is achieved by introducing *contextual* and *projection* oper-ions into topological logic (Rescher and Urquhart, 1971) (i.e. standard propositional gic added with a parameter operator $P\alpha$, where $P\alpha(p)$ is intended to mean "propo-ion p is realized at the position α"). The first identifies the domain or level of time anularity at which a given formula has to be considered. The second is used to con-ain formulae to different domains. The hierarchy of time is a linear structure called universe of domains, where a *granularity ordering* relationship is imposed over this iverse. Also a partial ordering of *disjointedness* is provided to relate domains at dif-ent levels of granularity. This is similar to the *partitioning scheme* of Dean's approach *. cit.*). Temporal domains are related to our concept of *modular temporal class*. Their ncept of locally temporally valid is related to the meaning of the throughout temporal nnective. Finally, because we define grains of time based on modular chain of modular s, cyclicity of events happening at different granularities is more easily obtained.

In the context of reactive systems specification and reasoning, Fiadeiro and Maibaum 994) proposes a hierarchical (vertical) decomposition (or abstract implementation), of ject specification in temporal logic. Such objects are seen as building blocks of the sign process of reactive systems. At each layer of such a hierarchy there is a logic aling with a single time scale, isomorphic to the set of natural numbers, and there a collection of objects that may be used for composing complex objects (systems)at her levels of abstraction. In this way, temporal execution of an abstract action is done the temporal execution of concrete actions of the level below. The interface between

both levels is given by axioms which says when the concrete actions start, are being executed or have finished. This work does not intend to represent or reason about time explicitly. However, close observation shows us that the granularity of time is embodied within the specification of actions at each level. We could not see how cyclical processes might be represented in such a framework. Although it is allowed to represent interaction between abstract and concrete actions, the opposite direction of relation does seem to be straightforward. Our approach, though based on explicit reference of time and different mechanisms, is more general because we allow interaction in both direction.

An approach which allows many granularities in the same logic, for specifying asynchronous execution of agents is proposed in Fisher (1995). In this work, granularity (though not mentioned) is achieved by providing each agent with its own local clock represented by the predicate $tick(O)$. The problems with the "next" operator is solved by using the auxiliary predicates $next\text{-}tick(O, X)$—which is true if X is satisfied within the next O tick, analogously $last\text{-}tick(O, X)$. No mechanism is proposed for dealing with interacting agents working at different ticks of the global clock.

The systems we have mentioned so far do not provide mechanisms for representing events, or actions at different levels of time granularity and cyclicity in the same logical framework. Only recently, in a parallel work to ours, Cukierman and Delgrand (1995) propose a framework of time based on the notion of *calendars* which are regarded as being cyclic temporal objects, and are related to our concept of MTC. Since TUs are formally represented in a linear hierarchy, recurrent activities are dealt with *non-convex* intervals as suggested in Ladkin's approach (*op. cit.*). While granularity is obtained by decomposing all TUs into contiguous partially ordered sequences of other TUs, we take a more abstract way by defining a chain of MTCs and thus obtaining both concepts at once. TUs are related to our temporal terms using a different notation, but the set-based language for specifying TUs generates complex expressions to be read when representing concepts like *collection intervals*. They do not explore mechanisms to obtain inferences about processes working at different time scales.

What seems to be common to almost all of these approaches is that they start from linear structure, and then try to achieve granularity by imposing a hierarchy among different time intervals, and cyclicity representation by using the concepts of *convex* and *non-convex*. Although we have not started from the same point, we could interpret our time intervals in the same way, but we do not need to do that for the understanding of the principles of the theory proposed here. Furthermore, the basic assumptions of these theories do not include cyclical aspects of time in their models. Such a need has also been addressed in Pachet *et al.* (1995) for dealing with musical objects.

Finally, the style in which we present our logic is very close to proposed in Fruhwirth (1996), where temporal reasoning is treated as an application of Annotated Constraint Logic Programming. The reason is because we also view a logical formula as a classical formula annotated with a PTE. Fruhwirth's work even uses a notation for time which is similar to our smallest interval. However, there are no special mechanisms for granularity of time and circular time as we have, although it seems to be possible.

7. Conclusion and Future Work

In this paper we presented a new theory of time granularity which can be easily understood and used to define as many levels of time hierarchy as needed. We also showed that such a theory is useful in the representation of cyclical processes in simulation models for

cosystems. In particular, the theory offers a simple and elegant mechanism to specify s many collections of time intervals as wanted, that is the concept of MTC.

As a specification language for simulation models which interact, the **NatureTime** gic was shown to be a powerful tool. It offers a very expressive way to define executable imulation models to be tested, mainly in the case of ecosystem domain.

Other possible branches of research from this work are:

To investigate the expansion of the logic for full resolution.

To adequate the temporal unification process for different types of temporal connectives. For instance, the present version would not deal properly with a temporal connective like "sometime".

To propose a more general proof procedure which can deal with PTE.

To extend to a multi-agent framework (Mota, 1995).

Acknowledgements

The first author would like to thank Wamberto Vasconcelos for his encouraging ideas uring many "academic coffees". We would like to thank Robert Muetzelfeldt and Paulo alles who have contributed to our understanding on ecological modelling and simulation. Ve also thank to Mandy Haggith for proof-reading this work and previous contribution hat helped the present paper. Finally, we thank the referees who carefully read and gave aluable comments for this final version.

References

en, J.F., Hayes, P.J. (1985). A common sense theory of time. In *Proc. IJCAI*, pp. 528–531.

en, J.F. (1983). Maintaining knowledge about temporal intervals. *Comm. ACM*, **26**(11).

en, J.F. (1984). Towards a general theory of action and time. *Artificial Intelligence*, **23**(1).

gs, N.L. (1987). *Discrete Mathematics.* Oxford Science Publication.

pessoni, E., Corsetti, E., Montanari, A., San Pietro, P. (1993). Embedding time granularity in a logical specification language for synchronous real-time systems. *Science of Computer Programming*, **20**(1):141–171.

ierman, D., Delgrand, J. (1995). A language to express time intervals and repetition. In *Proc. 2nd International Workshop on Temporal Representation and Reasoning*, Melbourne Beach, Florida, USA, April.

n, T. (1989). Using temoral hierarchies to efficietly maintain large temporal databases. *J. ACM*, **36**(4):687–718.

deiro, J.L., Maibaum, T. (1994). SOMETIMES "TOMOROW" IS SOMETIME action refinement in a temporal logic of objects. In *First International Conference on Temporal Logic*, pp. 48–66, Bonn, Germany, July. Springer-Verlag.

ber, M. (1995). Towards a semantics for concurrent metatem. In Fisher, M., Owens, R., eds, *Executable Modal and Temporal Logics*, pp. 86–102. Springer-Verlag. V(897).

hwirth, T. (1996). Temporal annotated constraint logic programming. *J. Symbolic Computation* **22**(5/6):555–583.

bbay, D., Reynolds, M. (1995). Towards a computational treatment of time. In *Handbook of Logic in Artificial Intelligence and Logic Programming*, volume 4 Epistemic and Temporal Reasoning. Oxford University Press.

bbay, D. (1989). The declarative past and imperative future. In Barringer, H., ed., *Coloquim on Temporal Logics and Specification*, pp. 409–448. Springer-Verlag. V(398).

gith, M., Robertson, D., Walker, D., Sinclair, F., Muetzelfeldt, R. (1992). TEAK—tools for eliciting agroforestry knowledge. In *British Computer Society Symposium of IT—Enabled Charge in Developing Countries.*

bs, J.R. (1985). Granularity. In *Proc. IJCAI*, pp. 1–4.

men, J.A.G.M. (1989). Reasoning about recurrence. Technical Report Technical Report 307, Department of Computer Science—University of Rochester, Rochester, USA.

kin, P. (1986a). Primitives units for time specification. In *Proc. AAAI*, pp. 354–359.

Ladkin, P. (1986b). Time representation: A taxonomy of interval relations. In *Proc. AAAI*, pp. 360–366.
Ladkin, P. (1987). The completeness of a natural system for reasoning with time intervals. In *Proc. AAAI*, pp. 462–467.
Leban, B., McDonald, D.D., Foster, D.R. (1986). A representation for collections of temporal intervals. In *Proc. AAAI*, pp. 367–371.
Montanari, A. (1994). A metric and layered temporal logic for time granularity, syncrony and asyncrony. Unpublished work of the First International Conference on Temporal Logic, July.
Mota, E. (1994). Temporal representation of ecological domains. DAI TP-31, Department of Artificial Intelligence, University of Edinburgh.
Mota, E. (1995). Time granularity in simulation models within a multi-agent system. DAI Discussion Paper 158, Department of Artificial Intelligence, University of Edinburgh, April.
Mota, E., Haggith M., Smaill, A., Robertson, D. (1995a). Time granularity in simulation models of ecological systems. In *Workshop on Excutable Temporal Logics—Montreal, Canada.* DAI RP-740, Edinburgh University.
Mota, E., Robertson, D., Muezelfeldt, R. (1995b). On the granular aspects of time in simulation models. TP-39, Detartment of Artificial Intelligence/University of Edinburgh.
Pachet, F., Ramalho, G., Carrive, J., Cornic, G. (1995). Representing temporal musical objects and reasoning in the muses system. In *International Congress on Music and Artificial Intelligence*, pp. 33–48.
Le Poidevin, R., MacBeath, M., eds (1995). *The Philosophy of Time*, chapter IX, pp. 149–167. Oxford Readings in Philosophy. Oxford University Press.
Rescher, N., Urquhart, A. (1971). *Temporal Logic.* Springer-Verlag.
Robertson, D., Bundy, A., Muetzefeldt, R., Haggith, M., Uschold, M. (1991). *Eco-Logic Logic-Based Approaches to Ecological Modelling.* The MIT Press.
Sinclair, F, Robertson, D., Muetzelfeldt, R., Walker, D., Haggith, M., Kendon, G. (1993). Formal representation and use of indigenous ecological knowledge about agroforestry. ODA Forestry and Agroforestry Research Strategy—Project R4731 Second Annual Report, University of Edinburgh.
Sterling, L., Shapiro, E. (1986). *The Art of Prolog.* The MIT Press.
van Emden, M.H. (1984). An interpreting algorithm for prolog programs. In *Prolog Implementations*, Ellis Horwood Series ARTIFICIAL INTELLIGENCE, pp. 93–110. Ellis Horwood Ltd.

Appendix A. Properties of $\overset{t}{\succ}$

In what follows, $\mathcal{C}_i^{mi}$ means the MTC of level i defined with modular value mi. This relation establish a sub-division relationship between MTCs. The $\overset{t}{\succ}$ relation has the following properties.

The $\overset{t}{\succ}$ relation has the following properties.

transitive—if $\mathcal{C}_i^{mi}$, $\mathcal{C}_j^{mj}$ and $\mathcal{C}_k^{mk}$ are MTCs and $\mathcal{C}_k^{mk} \overset{t}{\succ} \mathcal{C}_j^{mj}$ and $\mathcal{C}_j^{mj} \overset{t}{\succ} \mathcal{C}_i^{mi}$, then $\mathcal{C}_k^{mk} \overset{t}{\succ} \mathcal{C}_i^{mi}$.

reflexive—if $\mathcal{C}_i^{mi}$ is a MTC then $\mathcal{C}_i^{mi} \overset{t}{\succ} \mathcal{C}_i^{mi}$ (every MTC can be subdivided in itself)

anti-symmetric—if $\mathcal{C}_i^{mi}$, $\mathcal{C}_j^{mj}$ are MTCs, and $i \neq j$, and $\mathcal{C}_j^{mj} \overset{t}{\succ} \mathcal{C}_i^{mi}$, then $\mathcal{C}_i^{mi} \overset{t}{\not\succ} \mathcal{C}_j^{mj}$.

Appendix B. Up-Wave Modular Sum and Subtraction

DEFINITION B.1. *Let* $P = p(\Delta, c_i)$, *where* c_i *defines another MTC as* $mod\_temp\_class(c_{i+1}, c_i, m)$, *and* $I = t(s_1, \ldots, s_k)$. *The* up-wave modular sum *between* P *and* I, $I \,\omega_{\oplus}\, P$, *is defined as*

$$I \,\omega_{\oplus}\, P = t(s_1, \ldots, s_i + \Delta, \ldots, s_k), \text{ iff } s_i + \Delta < m$$

$I \,\omega_{\oplus}\, P = t(s_1, \ldots, s_i \oplus \Delta, \ldots, s_k) \,\omega_{\oplus}\, P'$, *iff* $s_i + \Delta \geq m$, *and* $P = p(\Delta', c_{i+1})$, *where* $\Delta' = s_i + \Delta$ *div* m.

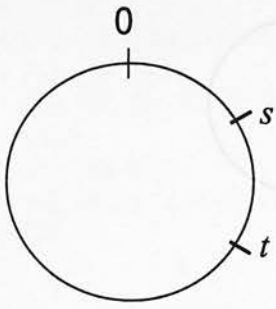

Figure 3. Time interval in a modular temporal structure.

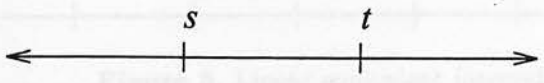

Figure 4. Correspondent interval at any arbitrary interval in the infinte line.

)EFINITION B.2. *Let $P = p(\Delta, c_i)$, and c_i defines another MTC as mod_temp_class$(c_{i+1}$, $_i m)$, and $I = t(s_1, \ldots, s_k)$. We call the* up-wave modular subtraction *between P and I, $\omega_\ominus P$, defined as*

$$I \,\omega_\ominus P = t(s_1, \ldots, s_i - \Delta, \ldots, s_k), \;\; \textit{iff } \Delta < s_i$$

$\omega_\ominus P = t(s_1, \ldots, s_i \ominus \Delta, \ldots, s_k)\, \omega_\ominus P'$, *iff* $\Delta \geq s_i$ *and* $P' = p(\Delta', c_{i+1})$, *where* $\Delta' = \Delta$ *iv* m.

B.1. DIAMETER FUNCTION

)EFINITION B.3. *Let S be an interval in a Linear-Cyclic hierarchy of time, P a period f time. We call* diameter of time, *written $\oslash$, to the function which maps S to P. More nerally, $\oslash : \mathcal{E} \rightarrow \mathcal{P}$, where $\mathcal{E}$ is the set of all temporal entities and $\mathcal{P}$ the set of all riods, or length of time.*

B.2. LINEAR REALIZABILITY OF A CIRCULAR TIME STRUCTURE

This section presents what was called *Linear Realizability* in Rescher and Urquhart 971) in which it was established a relationship between a course of history in a circular ructure of time and the same course in a linear time. Consider a temporal structure hich is one-dimension, finite and closed C^m, where m is the number of elements in the rcular set C. Then any of the possible courses of history realized in C can also be alized on the line. We can see this if we consider the arbitrary interval $s \ldots t$ in the rcle C^m as shown in Figure 3.

By putting it in correspondence with an arbitrary interval (with the same size), in the finite line, as shown in Figure 4.

Now, at any distance m forwards from s and t in the circular structure C^m in corre- ondence the forwards oints at the same distance from the linear interval $s \ldots t$; and alogously backwards. Thus we put the circle into correspondence over and over again th an *equally long* segment on the line as depicted in Figure 5.

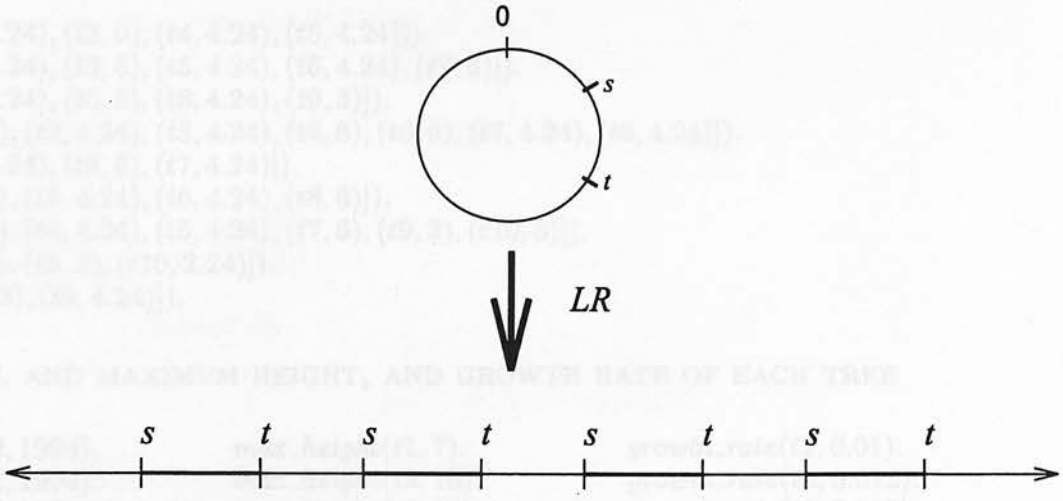

Figure 5. Linear equivalent intervals of the circular interval.

Appendix C. Matching Between Cyclical Intervals

DEFINITION C.1. *Given the cyclical intervals* $S = i(s_1 \ldots s_2, c_i)$ *and* $T = i(t_1 \ldots t_2, c_i)$ *of a MTC* c_i. *Then, the matching between these intervals results in* $i(t_1 \ldots t_2, c_i)$ *in the case that one of the following holds.*

$s_2 \geq t_2$ *and* $t_2 > t_1$ *and* $t_1 \geq s_1$, *or*
$t_2 > t_1$ *and* $t_1 \geq s_1$ *and* $s_1 > s_2$, *or*
$s_1 > s_2$ *and* $s_2 \geq t_2$ *and* $t_2 > t_1$, *or*
$t_1 \geq s_1$ *and* $s_1 > s_2$ *and* $s_2 \geq t_2$.

In this case we say that T *is* included *in (or is during)* S.

DEFINITION C.2. *Given the cyclical intervals* $S = i(s_1 \ldots s_2, c_i)$ *and* $T = i(t_1 \ldots t_2, c_i)$ *of a MTC* c_i. *Then, the matching between these intervals results in*

$i(t_1 \ldots s_2, c_i)$ *if one of the following holds*

$t_2 > s_2$ *and* $s_2 \geq t_1$ *and* $t_1 > s_1$, *or*
$s_2 \geq t_1$ *and* $t_1 \geq s_1$ *and* $s_1 > t_2$, *or*
$t_1 > s_1$ *and* $s_1 > t_2$ *and* $t_2 > s_2$, *or*
$s_1 > t_2$ *and* $t_2 \geq s_2$ *and* $s_2 \geq t_1$

$i(t_1 \ldots s_2, c_i)$ *and also* $i(s_1 \ldots t_2, c_i)$ *if one of the following holds*

$t_2 \geq s_1$ *and* $s_1 > s_2$ *and* $s_2 \geq t_1$, *or*
$t_1 > t_2$ *and* $t_2 \geq s_1$ *and* $s_1 > s_2$.

In both cases S *and* T *overlap.*

Appendix D. KB Definition for the Example 1

D.1. NEIGHBOURS DEFINITION

neighbours$(t1, [(t2, 4.24), (t3, 4.24), (t5, 6)])$.

$neighbours(t2, [(t1, 4.24), (t3, 6), (t4, 4.24), (t5, 4.24)]).$
$neighbours(t3, [(t1, 4.24), (t2, 6), (t5, 4.24), (t6, 4.24), (t7, 6)]).$
$neighbours(t4, [(t2, 4.24), (t5, 6), (t8, 4.24), (t9, 3)]).$
$neighbours(t5, [(t1, 6), (t2, 4.24), (t3, 4.24), (t4, 6), (t6, 6), (t7, 4.24), (t8, 4.24)]).$
$neighbours(t6, [(t3, 4.24), (t5, 6), (t7, 4.24)]).$
$neighbours(t7, [(t3, 6), (t5, 4.24), (t6, 4.24), (t8, 6)]).$
$neighbours(t8, [(t2, 6), (t4, 4.24), (t5, 4.24), (t7, 6), (t9, 3), (t10, 3)]).$
$neighbours(t9, [(t4, 3), (t8, 3), (t10, 2.24)]).$
$neighbours(t10, [(t8, 3), (t9, 4.24)]).$

D.2. INITIAL AND MAXIMUM HEIGHT, AND GROWTH RATE OF EACH TREE

$eight(t1, 1) @ t(14, 2, 1994).$	$max\_height(t1, 7).$	$growth\_rate(t1, 0.01).$
$eight(t2, 1) @ t(14, 2, 1994).$	$max\_height(t2, 10).$	$growth\_rate(t2, 0.012).$
$eight(t3, 1) @ t(14, 2, 1994).$	$max\_height(t3, 12).$	$growth\_rate(t3, 0.015).$
$eight(t4, 1) @ t(14, 2, 1994).$	$max\_height(t4, 5).$	$growth\_rate(t4, 0.0009).$
$eight(t5, 1) @ t(14, 2, 1994).$	$max\_height(t5, 5).$	$growth\_rate(t5, 0.008).$
$eight(t6, 1) @ t(14, 2, 1994).$	$max\_height(t6, 7).$	$growth\_rate(t6, 0.011).$
$eight(t7, 1) @ t(14, 2, 1994).$	$max\_height(t7, 12).$	$growth\_rate(t7, 0.015).$
$eight(t8, 1) @ t(14, 2, 1994).$	$max\_height(t8, 6).$	$growth\_rate(t8, 0.01).$
$eight(t9, 1) @ t(14, 2, 1994).$	$max\_height(t9, 4).$	$growth\_rate(t9, 0.006).$
$eight(t10, 1) @ t(14, 2, 1994).$	$max\_height(t10, 13).$	$growth\_rate(t10, 0.018).$

D.3. INFLUENCE OF OTHER TREES IN ONE TREE

$$
\begin{aligned}
&nfluences([], \_, []).\\
&nfluences([Y|T], Time, [R1|TR]) :-\\
&\quad ind\_influence(Y, Time, R1),\\
&\quad influences(T, Time, TR).\\
&nd\_influence((Tree, D), T, I) :-\\
&\quad solve(height(Tree, H) @ T, \_),\\
&\quad (I \text{ is } H/(D * 1000)), !.
\end{aligned}
$$

Appendix E. KB Definition for the Example 2

E.1. PROGRESS AND INFLUENCE SPECIFICATION

The progress observed of the value of an attribute *Att* of an agent *Obj*, from a given temporal ntity T during a given period of time P, is represented in a list composed with the values for *Att* during T. This list is computed as defined in the meta-language as follows.

$$
\begin{aligned}
&rogress(value(Att, Obj, V)@T, P, [V|R]) :-\\
&\quad change(Att, Proc),\\
&\quad scale(time, Obj, Proc, C),\\
&\quad solve(value(Att, Obj, V) @ T, \_),\\
&\quad future(T, P, Tf),\\
&\quad progression(value(Att, Obj, V) @ T, Tf, C, R).\\
&rogression(\_ @ T, Tf, C, []) :-\\
&\quad next(T, C, Tf).\\
&rogression(value(Att, Obj, Vi) @ Ti, Tf, C, [Vj|R]) :-\\
&\quad \neg\, next(Ti, C, Tf),\\
&\quad value(Att, Obj, Vj) @ p(1, C) \text{ after } Ti \Longleftarrow value(Att, Obj, Vi) @ Ti \,\&\, Constraints,\\
&\quad solve(Constraints, \_),\\
&\quad next(Ti, C, Tj),
\end{aligned}
$$

$progression(value(Att, Obj, Vj) @ Tj, Tf, C, R).$

$influence(GR, [], GR).$
$influence(GR, [Pos|R], RealGR) :-$
$\quad Pos > 7,$
$\quad NGR\ is\ GR - 0.02,$
$\quad influence(NGR, R, RealGR).$
$influence(GR, [Pos|R], RealGR) :-$
$\quad Pos \leq 7,$
$\quad influence(GR, R, RealGR).$

E.2. BUG'S POSITION

$new\_d(Pos1, M, H, D1, D2) :-$
$\quad Pos2\ is\ Pos1 + M,$
$\quad Pos2 \leq H,$
$\quad D2\ is\ D1 * (-1).$
$new\_d(Pos1, M, H, D1, D2) :-$
$\quad Pos2\ is\ Pos1 + M,$
$\quad Pos2 > H,$
$\quad D2\ is\ D1 * (-1).$